Univerzita Karlova v Praze

Pedagogická fakulta

BAKALÁŘSKÁ PRÁCE

2021 **Kamil Friš**

Univerzita Karlova v Praze

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

Rozvoj algoritmického myšlení žáků vyššího sekundárního vzděláváni

Development of algorithmic thinking of pupils higher secondary education

Kamil Friš

Vedoucí práce: PhDr. Petra Vaňková, Ph.D.

Studijní program: B7507 Specializace v pedagogice

Studijní obor: Informační technologie se zaměřením na vzdělávání

2021

UNIVERZITA KARLOVA
PEDAGOGICKÁ FAKULTA
Katedra informačních technologií a technické výchovy

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

akademický rok 2018/2019

Jméno a příjmení studenta: **Kamil Friš**

Studijní program: **B7507 Specializace v pedagogice**

Studijní obor: **Informační technologie se zaměřením na vzdělávání**

Název tématu práce v českém jazyce:
Rozvoj algoritmického myšlení žáků vyššího sekundárního vzdělávání

Název tématu práce v anglickém jazyce:
Development of algorithmic thinking of pupils higher secondary education

Stručná charakteristika tématu:
Cílem práce je analyzovat možnosti rozvoje algoritmického myšlení se zaměřením na vyhledávání chyb v algoritmech u žáků vyššího sekundárního vzdělávání se zapojením robotické programovatelné hračky.

Zásady pro vypracování:
- Na základě prostudovaných informačních zdrojů zmapujte problematiku algoritmického myšlení se zaměřením na vyhledávání chyb v algoritmu.
- Vytvořte vhodné úkoly pro vyhledávání chyb v algoritmech pro žáky vyššího sekundárního vzdělávání s využitím robotické programovatelné hračky.
- Realizujte pilotní ověření navržených úloh v edukační praxi.
- Shrňte výsledky práce a doporučení pro další praxi.

Předpokládaná struktura práce:
Úvod - Cíl práce - Výběr a popis metod - Teoretická a terminologická východiska - Příprava modelového řešení úloh a jejich pilotní ověření v edukační praxi - Výsledky a jejich hodnocení - Závěry – Seznam použitých informačních zdrojů – Přílohy

Seznam doporučené literatury:
Při řešení budou využívány primární a sekundárních informačních zdroje, včetně elektronických, dle tematické orientace práce.

Vedoucí bakalářské práce: **PhDr. Petra Vaňková, Ph.D.**

Oponent bakalářské práce: (nepovinná položka)

Předpokládaný rozsah bakalářské práce[1]: 40 n.m.s.

Datum zadání práce: **4. 7. 2019**

Předběžný termín odevzdání práce:[2] **prosinec 2019**

V Praze dne: 4.7.2019

.....
doc. PhDr. Vladimír Rambousek, CSc.
garant studia

Student(ka) stvrzuje podpisem převzetí zadání bakalářské práce.

V Praze dne: 9.9.2019

.....
podpis studenta/studentky

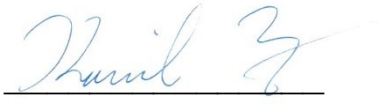
[1] Minimální rozsah bakalářské práce činí standardně 40 normostran (72 000 znaků vč. mezer) vlastního textu.
[2] Bakalářská práce je odevzdávána elektronicky prostřednictvím informačního systému dle harmonogramu akademického roku, zároveň se práce odevzdává v jedné tištěné podobě.

Odevzdáním této bakalářské práce na téma „Rozvoj algoritmického myšleni žáků vyššího sekundárního vzděláváni“ potvrzuji, že jsem ji vypracoval pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále potvrzuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

V Praze dne 12.7.2021

Kamil Friš

Rád bych touto cestou vyjádřil poděkování vedoucí práce PhDr. Petře Vaňkové, Ph.D. za cenné rady a trpělivost při vedení mé bakalářské práce.



Kamil Friš

ABSTRAKT

Bakalářská práce se zabývá rozvojem algoritmického myšlení žáků střední školy při výuce vyšších programovacích jazyků v předmětech informatiky. S pomoci robotické programovatelné pomůcky, jenž je programována s využitím vizuálního blokového programovacího jazyka typu Scratch, přináší příklady lekcí, které je možné zkombinovat s klasickou výukou programování. Dále přináší přehled programovatelných robotických hraček, které je možné začlenit do výuky na střední škole. V teoretické části se zabývá vymezením souvisejících pojmů, metod výuky a kompetencí žáka k předmětu programování.

KLÍČOVÁ SLOVA

Algoritmické myšlení, programování, práce s chybou, robotická programovatelná pomůcka, micro:bit

ABSTRACT

The bachelor's thesis deals with the development of algorithmic thinking for secondary school students in learning high-level programming languages in computer science lessons. With the help of a programmable robotics kit, which is programmed by using a block-based visual programming language like as Scratch, it provides examples of lessons that can be combined with the standard programming one. The thesis also provides an overview of programmable robot toys that can be used into secondary school teaching. The theoretical part deals with the definition of related concepts and teaching methods in the computer science lessons.

KEYWORDS

Algorithmic thinking, programming, error handling, programmable robotics kit, micro:bit

Obsah

Úvod 9

Cíl práce 10

Dílčí cíle 10

1 Výběr a popis metod 11

2 Teoretická a terminologická východiska 11

2.1 Informatické myšlení 11

2.2 Algoritmus 13

2.3 Algoritmizační kompetence 15

2.4 Algoritmické myšlení 15

2.5 Digitální kompetence 16

2.6 Chyby v algoritmech 17

2.6.1 Dělení chyb v programu 18

2.6.2 Příklady vyhledávání chyb v algoritmu 18

2.7 Výukové metody 22

2.7.1 Programované vyučování 22

2.7.2 Práce s chybou 23

2.7.3 Gamifikace výuky 24

2.7.4 Řízené objevování 25

3 Přehled robotických programovatelných pomůcek 27

3.1 LEGO Mindstorms 28

3.2 LEGO Education SPIKE Prime 31

3.3 Robotická stavebnice VEX 32

3.4 DJI RoboMaster S1 33

3.5 BBC micro:bit 34

3.5.1 Hardware micro:bitu 38

3.5.2 Software micro:bitu 39
3.5.3 Příslušenství k micro:bit 40
3.6 Arduino 41
3.7 Raspberry Pi 42
4 Modelové úlohy se zapojením robotické pomůcky 44
4.1 Výběr programovatelné robotické pomůcky 45
4.2 Syntaktické chyby micro:bitu a prostředí MakeCode 46
4.3 Popis jednotlivých lekcí s použitím micro:bit 47
4.3.1 Lekce 1 – semafor pro chodce 47
4.3.2 Lekce 2 – automobilový semafor 49
4.3.3 Lekce 3 – větvení a funkce 50
4.3.4 Další možné využití ve výuce 51
4.3.5 Lekce 4 – opravte chybu programu 52
4.4 Projekt robotického auta s důrazem na práci s chybou 53
4.4.1 Korekce nestejných parametrů motorů 54
4.4.2 Hardwarová chyba dálkového ovladače 56
4.4.3 Chyby měření sonaru 57
4.4.4 Další případy práce s chybou 58
5 Ověření navržených úloh v edukační praxi 60
5.1 Zpětná vazba z výuky 61
5.2 Zhodnocení výuky 63
Závěr 64
Seznam použitých informačních zdrojů 65
Přílohy 72
Příloha 1 – ukázka příkladu ze soutěže Bobřík informatiky 72
Příloha 2 – Pracovní list lekce 1 - semafor pro chodce 74

Příloha 3 – Pracovní list lekce 2 – automobilový semafor 75

Příloha 4 – Pracovní list lekce 3 – větvení a funkce 76

Příloha 5 – Pracovní list lekce 4 – opravte chybu v programu 77

Příloha 6 – Pracovní list lekce 5 - dálkově řízené vozítko 78

Příloha 7 – Fotodokumentace a videozáznamy 80

Úvod

Práce se zabývá rozvojem algoritmického myšlení žáků střední školy při výuce vyšších programovacích jazyků v předmětech informatiky. Má nalézt možnosti, jak zpestřit běžnou výuku programování a tím prohloubit algoritmizační schopnosti žáků. S pomocí robotické programovatelné pomůcky má ukázat jiný, názornější pohled na řešení algoritmů a vyhledávání případných chyb. Za pozitivní též považuji, možnost sestavování programů s využitím vizuálního blokového programovacího jazyka typu Scratch. To krom větší názornosti také umožňuje začlenit vytvořené příklady lekcí do klasické výuky jakéhokoli vyššího programovacího jazyka. Práce také přináší přehled programovatelných robotických hraček, které je možné začlenit do výuky na střední škole. V teoretické části se zabývá vymezením souvisejících pojmů, metod výuky a kompetencí žáka k předmětu programování.

Rozhodl jsem se v této práci zabývat zlepšením hodin programování. Také najít jinou názornější možnost, jenž by mohla pomoci žákovi, který bezradně kouká na svůj program v C#, JS či PHP, ve vývojovém prostředí, které mu příkazy našeptává, s možností používat další informační zdroje na celém internetu, vnuknout myšlenku, že v programu je asi třeba příkaz několikrát zopakovat a že na opakování máme cykly a že těch typů cyklů je více? Odpovědí mohou být hračky, robotické hračky, třeba když si budou žáci hrát, tak to pochopí. Hračky, které by byly při výuce programování dostupné a šlo by je začlenit do výuky, kdykoli by bylo potřeba.

V této práci se pokusím nalézt programovatelnou robotickou hračku, která by mohla sloužit jako stálá pomůcka pro výuku programování vyšších programovacích jazyků.

Cíl práce

Hlavním cílem práce je analyzovat možnosti rozvoje algoritmického myšlení se zaměřením na vyhledávání chyb v algoritmech u žáků vyššího sekundárního vzdělávání se zapojením robotické programovatelné hračky.

Dílčí cíle

Pro splnění hlavního cíle byly vymezeny následující dílčí cíle:

- Definovat koncept algoritmického myšlení a vymezit další související pojmy.
- Vytvořit přehled dostupných robotických programovatelných pomůcek vhodných pro žáky vyššího sekundárního vzdělávání.
- Vybrat vhodnou robotickou programovatelnou pomůcku použitelnou v hodinách stávajících předmětů zaměřených na výuku programování a pomoci tak rozvinout algoritmické myšlení žáků.
- Vytvořit vhodné úlohy zaměřené na vyhledávání chyb v algoritmech.
- Vytvořit vhodné úlohy pro žáky vyššího sekundárního vzdělávání s využitím robotické programovatelné pomůcky.
- Začlenit vytvořené úlohy do výuky programování a pomoci tak rozvinout algoritmické myšlení žáků.
- Ověřit vhodné úlohy v edukační praxi a následně shrnout.

1 Výběr a popis metod

V teoretické části byla použita práce s prameny, zejména s elektronickými, ale i z přednášek a analyticko-syntetickými postupy byly poznány a popsány základní pojmy a výukové metody.

Při vytváření a realizaci praktických příkladů byly zužitkovány teoretické poznatky a použity empirické metody, zejména experimenty, měření, pozorování výuky a dotazníkové šetření. Při přípravě lekcí a při výuce byla snaha uplatnit některé prvky výukových metod popsaných v kapitole 2.7. Reflexe formou pozorování žáků v hodině byla díky distanční výuce výrazně omezena.

Kritérii výběru robotické programovatelné pomůcky jsou: vhodnost pro vyšší sekundární vzdělávání, dostupnost, jednoduchost použití, cena, možnost tvořit program graficky pomocí bloků, možnost začlenění do stávající výuky, respektování náplně výuky ostatních předmětů, možnost pořídit si vlastní pomůcku. Jednotlivá kritéria jsou u každé pomůcky popsána v kapitole 3 Přehled robotických programovatelných pomůcek. Výběr konkrétní robotické programovatelné pomůcky je popsán v kapitole 4.1.

2 Teoretická a terminologická východiska

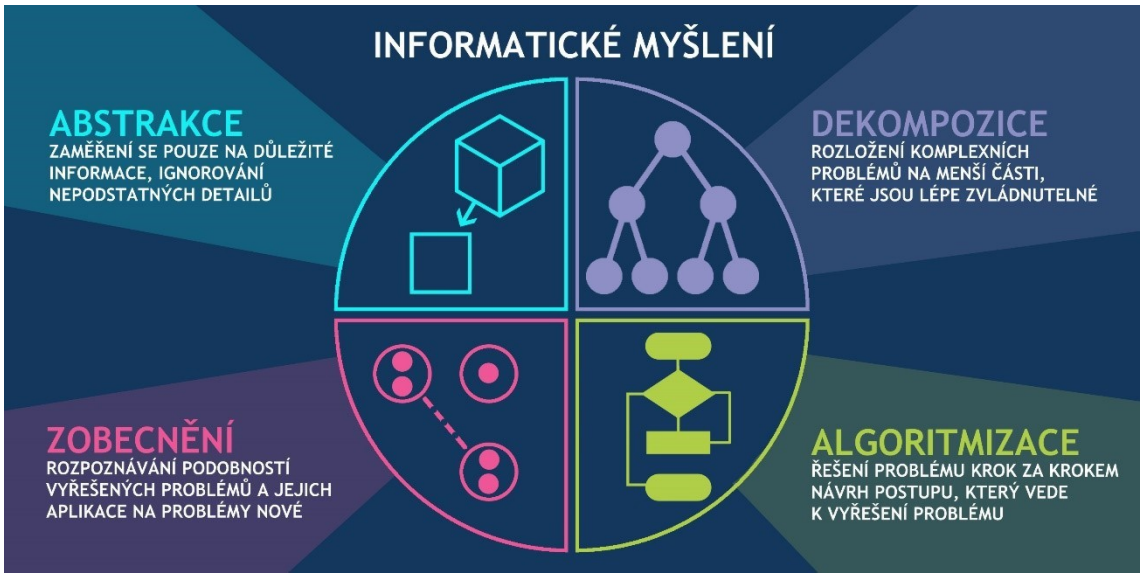
V této části se práce zabývá vymezením pojmů, které jsou pro práci klíčové: zejména informatické myšlení, algoritmické myšlení, algoritmizace a algoritmizační kompetence. Všechny tyto pojmy by se daly shrnout do termínu **digitální kompetence**.

2.1 Informatické myšlení

Termín informatické myšlení používáme pro ve světě používaný anglický termín computational thinking, jako nejvhodnější český termín, který by nezkresloval jeho význam a nebyl zavádějící. Součástí informatického myšlení je algoritmizace, schopnost hledat strukturu věcí a jevů a jejich popis, porozumění, jak jsou informace v počítači reprezentovány, řešení problémů, kódování, optimalizace a další. Informatické myšlení nemá téměř nic společného s uživatelskou obsluhou počítače a dalších technologií, neb takové používání počítače informatické myšlení nerozvíjí. [1]

Termín informatické myšlení, resp. computational thinking je novější a chybí všeobecná shoda na definici. Nejčastěji je informatické myšlení spojováno s těmito dovednostmi:

- ***Abstrakce*** *– zaměření se pouze na důležité informace, ignorování nepodstatných detailů.*
- ***Zobecnění*** *– rozpoznávání podobností vyřešených problémů a jejich aplikace na problémy nové.*
- ***Dekompozice*** *– rozložení komplexních problémů na menší části, které jsou lépe zvládnutelné.*
- ***Algoritmizace*** *– řešení problémů krok za krokem, nebo návrh postupu, který vede k vyřešení problému. [2]*



Obrázek 1: Informatické myšlení (zdroj: [2])

Podle mezinárodní společnosti Society for Technology in Education (ISTE) a Computer Science Teachers Association (CSTA) zabývající se standardy a inovacemi ve vzdělávání jsou pro informatické myšlení důležité tyto kompetence:

- Formulovat problémy takovým způsobem, aby bylo možné je vyřešit pomocí počítače a dalších nástrojů.
- Logicky organizovat a analyzovat data.
- Reprezentovat data pomocí abstrakce, jako jsou modely a simulace.
- Automatizovat řešení pomocí algoritmického myšlení (řadou přesně definovaných kroků).
- Identifikovat, analyzovat a implementovat možné řešení s cílem dosáhnout co nejefektivnější kombinace kroků.

- Zobecnit a převést problém pro vyřešení široké škály obdobných problémů. [3]

Při zvládání tohoto procesu by měl žák:

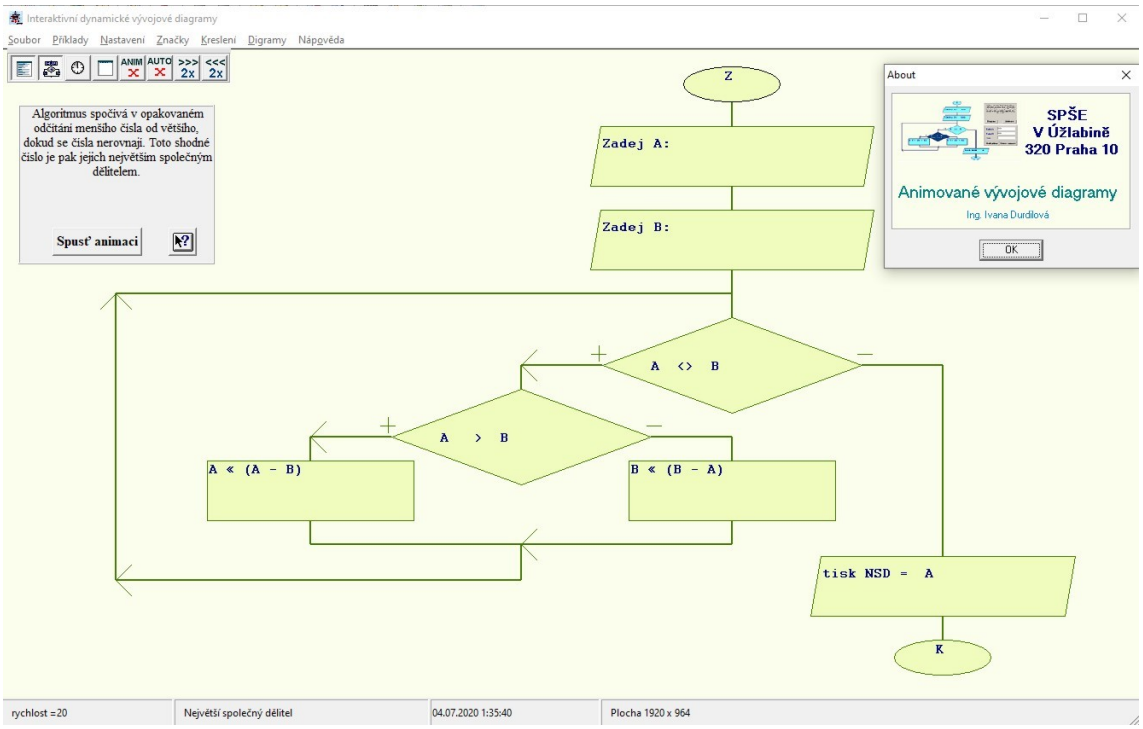
- *najít vnitřní jistotu při řešení problémů,*
- *vytrvat při řešení složitějších problémů,*
- *vypořádat se s nejednoznačně formulovanými problémy,*
- *být schopný řešit otevřené problémy,*
- *umět komunikovat a spolupracovat s ostatními spolužáky při řešení společných problémů. [4] [3]*

2.2 Algoritmus

Algoritmus je v dostupných zdrojích definován s drobnými odlišnostmi a v rozličné složitosti, např. Gerald Futschek užívá tuto jednoduchou definici: „Algoritmus je metoda k vyřešení problému, která se skládá z přesně definovaných pokynů“ [5].

Nebo jiná obšírnější definice: *Algoritmus je v podstatě posloupnost příkazů (pokynů či instrukcí), které po vykonání vedou k určitému cíli. Lze jej zapisovat buď slovně – běžným jazykem (kuchařské recepty, montážní návody), algoritmickým jazykem, programovacím jazykem, graficky (obrázkové návody, origami), nebo pomocí vývojových diagramů. Při zápisu algoritmu je třeba si uvědomit, kdo bude algoritmus vykonávat. V případě člověka stačí běžný slovní popis, pro počítač je třeba použít příslušný programovací jazyk. [2]*

Lze také definici zjednodušit na naprosté minimum bez cizích slov, které žákům nemusí být zcela jasné: „Algoritmus je (úplný,) jednoznačný, srozumitelný popis postupu, který vede k vyřešení všech úloh stejného typu“. I takto zkrácenou definici mohou mít někteří žáci problém si zapamatovat a pochopit. Je možné doplnit, že algoritmus je „návod“ a je vhodné ukázat, návod na sestavení nábytku, či stavebnice, kde jsou vlastně jen obrázky očíslované v daném pořadí. Tím se dostáváme k tomu, že algoritmus lze nakreslit obrázky, nebo standardizovanými značkami vývojového diagramu, slovním zápisem, (např. kuchařským receptem), nebo zápisem v programovacím jazyku.



Obrázek 2: Příklad zápisu algoritmu pomocí vývojového diagramu (zdroj: [6])

Algoritmy se řídí všechny technologie kolem nás, nejen pračky, myčky, ale i poměrně složitá zařízení jako mobilní telefon, či počítač. I lidská činnost obsahuje též algoritmy, např. vaření, cesta do školy, či práce. Tyto činnosti, lze rozdělit na jednotlivé kroky, které když učiníme ve správném pořadí, vedou k cíli.

Algoritmem by měl být nazýván pouze takový postup, který má určité vlastnosti:

- **Konečnost –** (neplést s rezultativností) algoritmus je časově omezený, po určitém počtu kroků skončí.
- **Rezultativnost –** vede vždy k cíli. Zadám-li vstupní data, algoritmus vždy vrátí odpověď, i chybová hláška je odpověď.
- **Determinovanost –** každý krok algoritmu je jednoznačně definován. Při stejných vstupních datech algoritmus probíhá stejně, je znám další krok.
- **Hromadnost** – řeší všechny úlohy daného typu lišících se vstupními údaji. Např. algoritmus pro výpočet kořenů kvadratické rovnice.

Algoritmy se sestávají z posloupnosti příkazu – **sekvence**, rozhodování – **větvení** a opakování – **cyklů**.

Způsobilost sestavit algoritmus (postup, jak problém řešit), který splňuje všechny tyto podmínky, a dovednost zapsat jej správně pomocí vývojového diagramu či v nějakém

programovacím jazyce vyžaduje algoritmické myšlení neboli schopnost algoritmizace. [2]

2.3 Algoritmizační kompetence

Na své přednášce „Výuka algoritmizace patří především do informatiky“, celostátní konference učitelů základních a středních škol – Počítač ve škole 2016, vyjmenoval J. Vaníček algoritmizační kompetence žáků takto:

- *Následovat (použít) algoritmus*
- *Objevit (najít) algoritmus*
- *Rozeznat cílový stav*
- *Rozeznat počáteční stav*
- *Ladit (odstraňovat chyby), testovat*
- *Porovnat algoritmy:*
    - *spolehlivost, náklady, rychlost, obecnost*
- *Posoudit problémy reálného světa*
    - *Z hlediska jejich řešitelnosti algoritmem [1]*

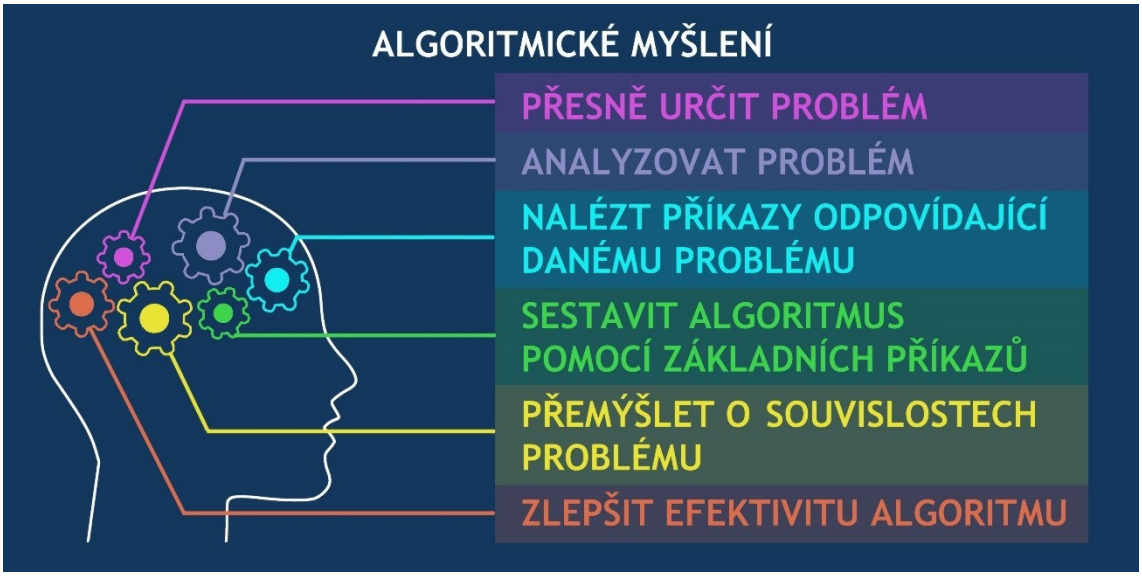
2.4 Algoritmické myšlení

Algoritmické myšlení jsou procesy, myšlenky, schopnosti, kterými vytváříme algoritmus (algoritmizujeme), či algoritmu zcela porozumíme, opravíme, nebo jej zefektivníme. Algoritmické myšlení je tedy součástí, podmnožinou informatického myšlení.

Například G. Futschek (2006) jej definuje jako soubor schopností, které jsou spojeny s vytvářením a porozuměním algoritmu:

- schopnost analyzovat dané problémy
- schopnost přesně určit problém
- schopnost najít základní akce, které jsou přiměřené danému problému
- schopnost vytvořit správný algoritmus k danému problému pomocí základních příkazů
- schopnost přemýšlet o všech možných zvláštních a běžných případech problému
- schopnost zlepšit efektivitu algoritmu [5]

Algoritmické myšlení má dle G. Futschek (2006) silný tvůrčí aspekt. Pokud někdo chce vytvářet nové algoritmy, které řeší dané problémy, potřebuje schopnost algoritmického myšlení. [5]



Obrázek 3: Algoritmické myšlení (zdroj: [2])

Algoritmické myšlení je jednou z digitálních kompetencí žáků, na jejíž rozvoj je v posledních letech kladen po celém světě velký důraz. Algoritmické myšlení žáků ovlivňuje mnoho dílčích kognitivních faktorů, bez nichž není možné takové myšlení dále rozvíjet. Jedná se zejména o schopnost abstraktního a logického myšlení, myšlení ve strukturách, schopnost řešit problémy a kreativitu. [4]

2.5 Digitální kompetence

V České republice, dle připravované revize rámcového vzdělávacího programu v oblasti ICT, dochází k přijetí evropského rámce digitálních kompetencí DigCompEdu, a tedy k rozšiřování rozvoje algoritmického myšlení jako digitální kompetence do nižšího sekundárního vzdělávání, konkrétně do předmětů z oblasti ICT. [4]

Avšak stávající RVP pro základní vzdělávání, byť zmiňuje termín algoritmické myšlení, se právě a jen zaměřuje na uživatelskou obsluhu počítače a na práci s informacemi [7]. Z vlastní zkušenosti usuzuji, že v praxi ještě situace na školách mnohde horší. Informatické předměty mají nízkou hodinovou dotaci. Žáci přicházející na střední školy nemají potřebné kompetence ani v obsluze počítače, např. na úrovni základních modulů ECDL [8]. Věřím, že se situace v českém školství zlepší zejména po opatřeních souvisejících s onemocněním COVID-19, kdy učitelé a žáci skokově získali dovednosti spojené

s online vyučováním. Také v aktuální revizi RVP pro ZV již je kladen důraz na digitální kompetence.

2.6 Chyby v algoritmech

Je jasné, že čím složitější bude problém, který máme algoritmizovat, tím pravděpodobněji v něm uděláme chybu[1]. Schopnost odstraňovat chyby je jednou z algoritmizačních kompetencí zmíněných v kapitole 2.3. U rozsáhlých projektů bývá nalezení chyb a odladění programu velmi náročné. Ve výuce, ale není vždy časový prostor pracovat na náročných komplexních projektech. Žáci tak nemají možnost tuto algoritmizační kompetenci řádně procvičit a být tak připraveni do praxe.

Oporu k výuce programování pomocí rozsáhlejších projektů, které nutně přináší důkladnější orientaci na řešení chyb, lze nalézt např. u R. Pecinovského z článku „Tvorba učebnic a kurzů programování“ [9], kde hovoří o časnějším vštípení zásad návrhu architektury.

Z toho ovšem vyplývá, že by výklad neměl být postaven na jednoduchých AHA příkladech, ale že bychom měli se studenty od počátku pracovat na nějakém rozsáhlejším projektu, k němuž budeme přidávat jednoduché části případně některé z jeho částí upravovat. Tak se studenti současně naučí pracovat v režimu, který je mnohem bližší tomu, s nímž se setkají ve své pozdější praxi. [9]

Problémům s nedostatečným procvičením ladění programu a hledáním chyb lze předejít i tak, že důležité části programu předložíme žákům s náležitým výkladem. Aby se výuka nezúžila na prostý frontální výklad daného programového kódu, lze vnést do kódu určitou zejména sémantickou chybu, která může sloužit učiteli, jako indikátor, jak důkladně žáci daný problém pochopili. Obvykle některý žák v průběhu výkladu kód reklamuje, nestane-li se tak, je prostor na zadání úlohy na nalezení a odstranění chyby. Uplatní se tak i žáci, kteří by tak složitý programový celek nezvládli sami naprogramovat, ale zorientovat se v již předloženém (cizím) projektu a odladit jej, dokáží.

[1] V praxi učitele programování a tvorby webových aplikací na střední škole považuji za smysluplné, aby žáci tvořili komplexní složité programy, projekty. To se ale neobejde bez toho, aby žáci v programu udělali chyby, zejména sémantické a běhové. Považuji za důležité, aby učitel dával žákovi průběžnou zpětnou vazbu a nasměroval ho na vyřešení problému.

2.6.1 Dělení chyb v programu

Při programování jakékoli aplikace může dojít k několika typům chyb:

- **Syntaktické chyby –** jsou to chyby gramatické: překlep, chybný název příkazu, zapomenutý znak, nejčastěji středník. Tyto chyby lze detekovat automaticky. Dnešní programovací prostředí (IDE) vyznačují syntaktické chyby prakticky ihned, ba dokonce jim předcházejí, tzv. IntelliSense nápovědou, která programátorovi nabízí k dokončení použitelné příkazy. Obvykle je syntaktická chyba v editoru ihned červeně podtržena a v programovacím prostředí lze zjistit i popis chyby, jenž napomůže k jejímu opravení. Program se syntaktickou chybou obvykle nelze spustit, respektive u skriptovacích jazyků program spustit lze, ale u syntaktické chyby dojde k zastavení programu a obvykle k vypsání příslušné chybové hlášky.
- **Sémantické chyby –** jsou chyby ve významu. Program se nechová, jak by měl, např. místo sčítání, odčítá. Tyto chyby nelze automaticky detekovat. Program musíme otestovat, vyzkoušet. Dále můžeme v některých IDE program ladit: můžeme jej spustit příkaz po příkazu (krokování, trasování) nebo zastavit na určitém místě.
- **Chyby za běhu programu –** na tyto chyby nemá programátor přímo vliv. Příkladem, může být, že uživatel zadá nečíselnou hodnotu do vstupu, kde program očekává číslo. Těmto chybám musí programátor předcházet. V objektově orientovaných jazycích lze tyto chyby řešit výjimkami.

Při výuce programování je třeba na řešení těchto chyb dbát a věnovat se jim, neb je to jedna z algoritmizačních kompetencí.

2.6.2 Příklady vyhledávání chyb v algoritmu

Na své přednášce „Výuka algoritmizace patří především do informatiky“, celostátní konference učitelů základních a středních škol – Počítač ve škole 2016, prezentoval J. Vaníček příklad viz. Obrázek 4.

Tečky na kostce – určení správného algoritmu

Robot v továrně na hrací kostky používá pouze 3 příkazy na vykreslení teček: *draw_1, draw_2* a *draw_2A*.

Na obrázcích je vidět, jak každý z nich funguje:

V programu lze použít ještě příkaz *turn_90*, který otočí kostku o 90°. Kombinováním všech příkazů můžeme z teček kreslit různé obrazce. Například postupným provedením příkazů *draw_1, draw_2, turn_90* vznikne tento obrazec:

Která sekvence příkazů vykreslí následující obrázek?

Odpovědi:

1. draw_2, draw_2A, turn_90, draw_2
2. draw_2A, draw_2, turn_90, draw_2
3. draw_2A, turn_90, draw_2, draw_1
4. draw_2, turn_90, draw_2, draw_2A

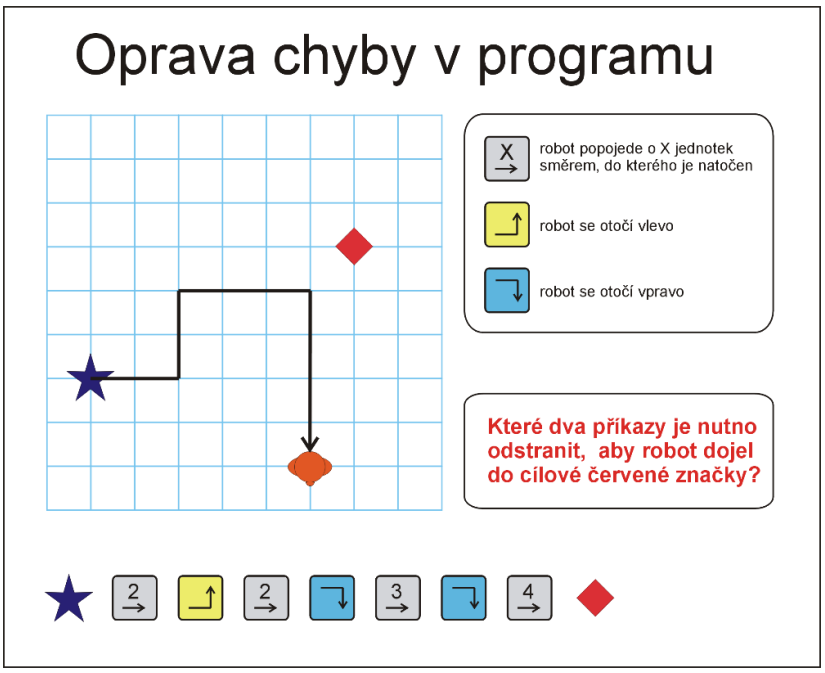
V této úloze pro středoškoláky žák musí rozhodnout, který z algoritmů vede k danému cílovému stavu.

Obrázek 4: příklad na vyloučení chybných algoritmů (zdroj: [1])

U této situační úlohy, Tečky na hrací kostce, máme nalézt správný algoritmus. Úlohu lze řešit tak, že si stanovíme dodatečná pravidla, která nesmí algoritmus obsahovat. Např. pro obrázek šestky nesmí algoritmus obsahovat příkaz „draw_1“. Tím vyloučíme chybný algoritmus č. 3. Dále nesmí být použit příkaz „draw_2A“ před „turn_90“, čímž stanovíme, že algoritmy č. 1 až 3 jsou chybné. Správnost algoritmu č.4 již stačí pouze ověřit. Takto jednoduché příklady algoritmu je možné analyzovat bez rozkreslování dílčích kroků na papíře, cvičí se tak představivost. Příklad tak může být předán žákům formou digitalizovaného samoopravného testu. Žákům, kteří odpovědí chybně, je však nutno dát zpětnou vazbu a řešení jim vysvětlit. Příklad není postaven na konkrétním programovacím jazyku. Čili tento typ úloh je vhodný do úvodních hodin programování na střední škole.

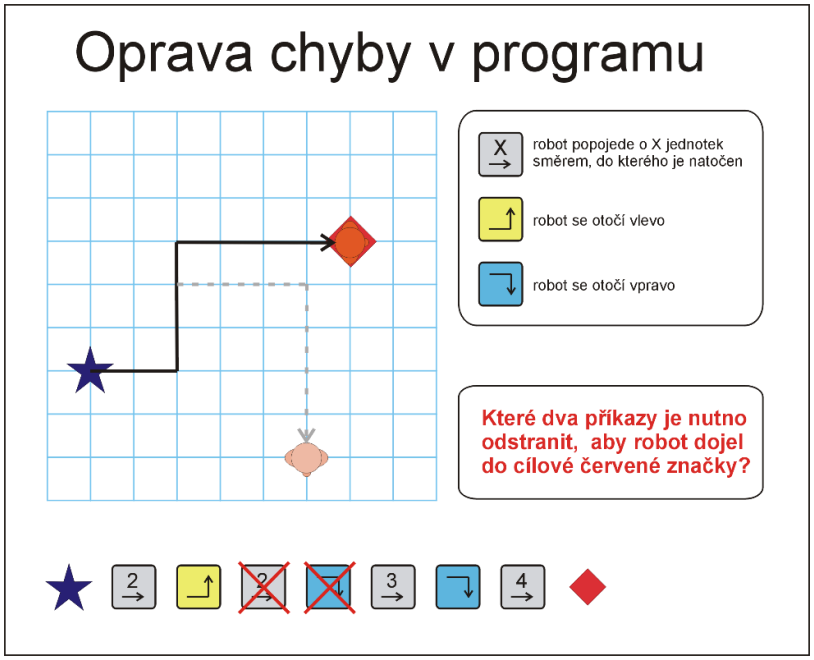
Dále J. Vaníček na své přednášce uvádí příklad s chybně naprogramovaným robotem „číšníkem“, který nedojde k určenému stolu, jenž má obsloužit. Žák nemá vymyslet nový správný algoritmus, má odebrat právě dva příkazy z chybného algoritmu, a tak jej opravit. [1] Je tedy třeba plně prozkoumat algoritmus a vyzkoušet varianty, které po odebrání dvou příkazů vedou k cíli. Též je možné polemizovat nad tím, že i takto opravený algoritmus by šel ještě zjednodušit, respektive zvolit jinou jednodušší cestu robota a použít tak méně příkazů, čímž by se snížilo riziko udělat onu prvotní chybu v algoritmu. Nicméně primárním cílem úkolu je, aby žák analyzoval a opravil daný algoritmus. Příklad

je opět oproštěn od konkrétního jazyka. Čili není zatížen potřebou nejprve probrat syntaxi programovacího jazyka.



Obrázek 5: Úloha na opravu chybně zadaného algoritmu (převzato z přednášky [1])

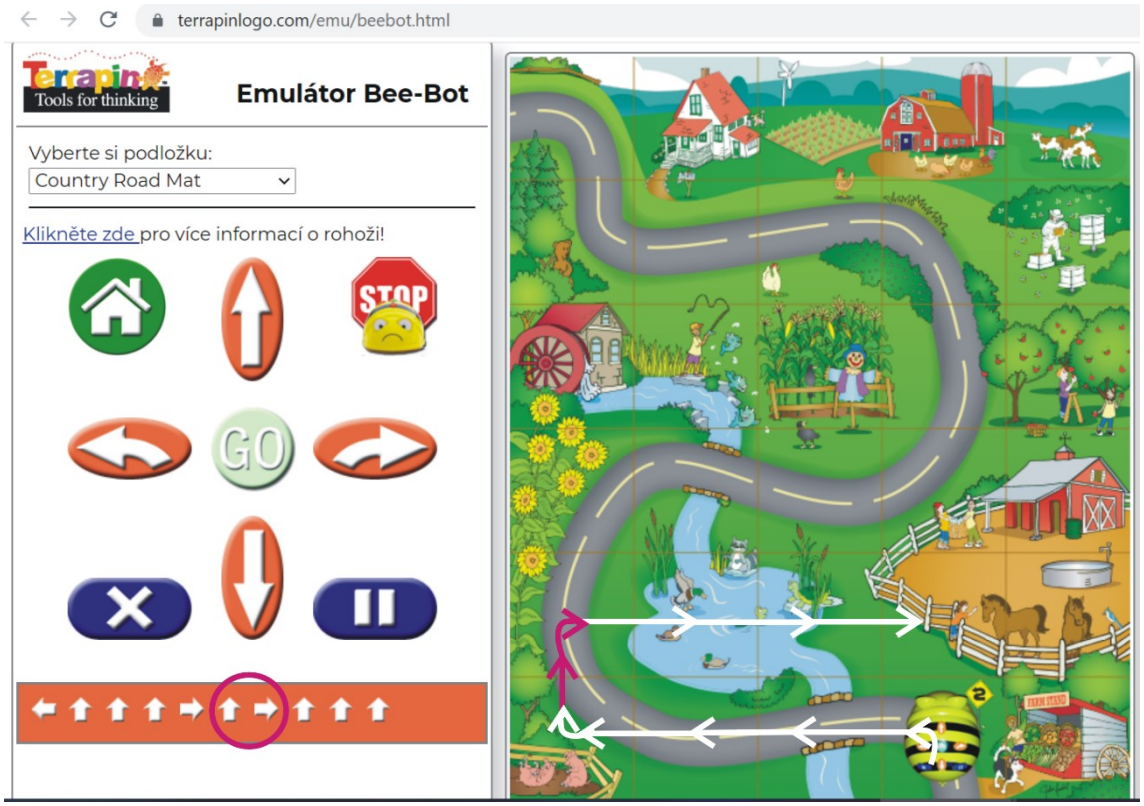
Žák 1. a 2. ročníku střední školy by měl dovodit správné řešení, tedy že první otočka vpravo (čtvrtý příkaz) je o políčko dříve a vůbec v algoritmu nemá být, tím pádem je třeba zrušit i předchozí příkaz posunu o dvě pole. Následně je třeba prověřit správnost opraveného algoritmu.



Obrázek 6: Řešení úlohy s chybně zadaným algoritmem (převzato z přednášky [1])

Tento princip úkolu lze demonstrovat i pomocí výukových pomůcek – programovatelných robotů. Lze jej modifikovat i pro žáky základní školy. Vhodným robotem pro tento typ úloh může být *Cubetto* [10], nebo *Bee-bot* [11]. Nemáme-li k dispozici příslušného robota, lze využít např. online emulátor (Emulátor pro Bee-Bot je dostupný na: https://beebot.terrapinlogo.com/.)

V tomto příkladu (Obrázek 7) se má včelka dostat ze své výchozí pozice do mlýna. Podle chybně zadané posloupnosti příkazů však dojede oklikou do hřebčína. Žák má opět najít právě dva příkazy, které je nutné odebrat, aby včelka dojela do mlýna.



Obrázek 7: Příklad chyby v algoritmu v emulátoru Bee-bot (použit emulátor na: https://beebot.terrapinlogo.com/)

Další příklad je kompletně převzat ze soutěže Bobřík informatiky z roku 2019 kategorie Junior [12], který mají řešit žáci 1. a 2. ročníku střední školy. Je součástí přílohy (Příloha 1), kde má žák naopak najít, který výstup z algoritmu, na výrobu náramků, je chybný. V algoritmu se dokonce vyskytuje i rekurze. Algoritmus je zadán graficky, bez příkazů v konkrétním programovacím jazyce. Součástí druhé strany přílohy je i řešení a vysvětlení. Přístup k této zpětné vazbě žák získá po absolvování celého testu.

2.7 Výukové metody

Výukové metody zejména v předmětech informatiky a programování mají, oproti ostatním předmětům, větší potenciál využívat digitální nástroje, jak pro výuku, tak pro automatizované ověřování znalostí žáků. Je třeba ale počítat s tím, že příprava a nastavení takovýchto výukových prostředí je časově náročná.

V poslední době se také často hovoří o začlenění robotických systémů do výuky, nakonec je ale učitel ten, který si příslušný robotický systém musí osvojit, zvolit vhodnou výukovou metodu a aplikovat nově pojatou výuku s robotickými systémy do již zaběhnuté výuky. To nemusí být vždy jednoduché, což potvrzuje např. Martina Kupilíková, ředitelka Centra robotiky v Plzni a specialistka na vzdělávání učitelů, v článku „Co může robotický systém přinést do výuky?“ v časopise e-Mole.cz [13]:

„To, že roboti přinesou učitelům víc práce. Je jasné – minimálně, než se pedagog naučí s danou technologií pracovat, jak po odborné, tak i po metodické stránce. Ale výuka programování a robotiky může mít veliký přínos. Dovoluje u žáků rozvíjet to, co je potřebné pro běžný život – algoritmické myšlení, kreativitu, systematický přístup, manuální zručnost, představivost, schopnost řešit problémy a spolupracovat,“ … [13]

„Ve výuce je ale podstatný lidský faktor – tedy pedagog. Technologie, kdyby byla sebezajímavější, za nás tu práci neudělá. Jde především o smysluplné využití té dané technologie.“ … „Každopádně už nějakou dobu v souvislosti s technickým vzděláváním slýcháme z různých stran termíny jako PRŮMYSL 4.0, PRÁCE 4.0, VZDĚLÁNÍ 4,0, PRŮMYSLOVÁ REVOLUCE 4.0“… [13]

Některé prvky z níže uvedených výukových metod jsou zakomponovány do praktických příkladů uvedených v kapitole **Chyba! Nenalezen zdroj odkazů.**.

2.7.1 Programované vyučování

Programované učení je vyučovací metoda založená na řízení učebních procesů žáků. Vychází z behaviorismu a neobehaviorismu a ze základního vzorce S – R (stimul – reakce), který zde má podobu U – Z (učení – zpevnění). Vznik této metody je spojen se jmény Sidney L. Pressey a B. F. Skinner a první polovinou 20. století. [14]

Metoda je založena na **principu malých kroků**, výukové látky, která na sebe bezprostředně navazuje a kterou si žák dávkuje **vlastním tempem**. Po každém výukovém

kroku následuje testování aktivní měřitelnou odpovědí žáka, s jejíž správností je žák seznámen. V případě neúspěšné odpovědi je žák vrácen k procvičení o krok zpět. [14]

V návaznosti na rozvoj algoritmického myšlení poskytuje programované učení žákům velkou míru individualizace výuky. Každý může postupovat programem vlastním tempem a v případě potřeby se vždy vrátit zpět a znovu se zamyslet nad řešeným problémem. [4]

Aplikované prvky této metody lze shledat v různých moderních výukových tutoriálech. Za zdařilé považuji kurzy na CODE.org, například „Hodinu kódu s Minecraftem“ (https://code.org/minecraft) [15], kde v grafickém programovacím prostředí typu Scratch řeší žák v jednotlivých krocích drobné úkoly. Motivačním prvkem pro žáky je zde též gamifikované prostředí oblíbené hry. Tento typ kurzu lze doporučit pro 1. stupeň základní školy.

Tato metoda programovaného učení řeší diferenciaci pracovního tempa žáků, což je v dnešním českém školství s debatami o inkluzi tématem dne.

2.7.2 Práce s chybou

Při výuce algoritmizace je práce s chybou běžnou součástí vývoje aplikace. Již v kapitole 2.6.1 Dělení chyb v programu, jsem zmínil, že na některé typy chyb vývojové prostředí aplikací ihned reaguje a poskytuje žákovi okamžitou zpětnou vazbu. Některé chyby je ale těžké odhalit, což může být i nad síly žáka. Učitel má v tomto případě nespornou výhodu, větší zkušenosti a také se velmi pravděpodobně setkal již se stovkami obdobných chyb, které udělali předchozí žáci. Občas se vyskytne žák, který věnuje zbytečně neúměrně velké množství času na nalezení chyby v programu, místo aby se zeptal vyučujícího, který je schopen mu ihned poradit. Zarážející je, že nemalé procento žáků si nevzpomene na možnost vyhledat řešení aktuálního problému na internetu, či v nápovědě. Tyto informační zdroje je třeba žákům při výuce povolit, až nařídit používat. V počítačové učebně to není problém. S určitými restrikcemi lze tyto zdroje povolit i u skládání testů a zkoušek.

Během své praxe jsem se setkal i s žáky, kterým algoritmické myšlení příliš nešlo, přesto se je podařilo namotivovat pro práci na programu, ten ale tvořili tzv. „metodou slepých uliček“. Je otázkou, zda vývoj aplikace takovouto hrubou silou spadá do vhodných metod pro programování, byť jí lze dosáhnout dílčích výsledků. Chyb dělá žák už ale moc a nemá šanci se z nich poučit.

Důležitá pro práci s chybou je zpětná vazba a řešení chyb je členěno na tyto fáze:

- **Nalezení chyby –** což může udělat program, vyučující, nebo žák samotný.
- **Identifikace chyby –** bližší specifikace chyby, umístění, závažnost.
- **Vysvětlení chyby –** příčiny vzniku, možné následky, pochopení chyby, aby nedošlo pouze k opravení následků chyby
- **Korekce chyby** – uplatnění poučení ze zpětné vazby a oprava chyby [4]

Z výše uvedeného plyne, že je třeba přistupovat k žákům individuálně a moderovat jejich proces opravování chyb.

2.7.3 Gamifikace výuky

V gamifikaci výuky jde především o využívání herních principů a prvků v mimoherním, tedy výukovém prostředí. V prostředí marketingu se tato metoda stala trendem již v 90. letech 20. století. Jde zejména o různé věrnostní programy, sbírání bodů a získávání odměn. Vše za účelem, aby obchodní společnosti motivovaly zákazníka k nákupu. Až počátkem 21. století se gamifikace dostává do edukačního prostředí, zejména kvůli snaze motivovat žáky. [4]

Motivace může být dvojího druhu. **Vnitřní** motivace je hnána touhou po poznání a sebenaplnění, naproti tomu pro **vnější** motivaci potřebuje člověk vnější stimul, tedy odměnu.

Princip aplikovaný ve hrách, kde postupným zvyšováním obtížnosti je hráč stimulován k plnění složitějších úkolů, je shodný s edukační praxí, tedy „od jednoduššího ke složitějšímu“. Zdá se být vhodné tyto dvě prostředí propojit.

Constance Steinkuehler Squire, docentka z University of Wisconsin-Madison, zabývající se propojením her se vzděláváním říká: „Kolem hry se vytvářejí komunity, které vykonávají ohromné množství intelektuální práce, a když jsou hotovy s jednou hrou, přesunou svoji pozornost k další, těžší. Dovedete si představit, že by se nám tohle podařilo nastolit i ve vyučování?“ [16]

Tento entuziasmus je vhodné trochu mírnit. Volba hry a komunity podléhá „módním“ trendům. Je potřeba nezapomínat na herní závislosti, což by se mohlo v lepším případě, v komplexně gamifikovaném výukovém prostředí, projevit tak, že žák by se specializoval na jeden předmět a ostatní opomíjel.

V dnešní době existují rozmanité nástroje pro gamifikaci výuky. Aplikace jako *Socrative* [17], *Kahoot!* [18], nebo do češtiny lokalizovaná *Toglic* [19] umožňují vzdělávat, či jen žákům zadávat úkoly formou zábavných miniher, křížovek a kvízů. Aplikace jako *Duolingo* [20] odměňuje účastníky přehršlí bodů a odznaků, kontroluje denní cíle, případně za odměnu odemyká nové lekce. Aplikace jako *Classcraft* [21], či *Class-Dojo* [22] umožňují vyučujícímu částečně přenést učební proces do formy hry. Žáci jsou zde motivování k plnění úkolů získáváním vylepšení pro svého avatara. Své úspěchy mohou sdílet a vzájemně se hodnotit. [4]

Gamifikovaným příkladem vhodným pro výuku informatiky je i v předchozí kapitole 2.7.1 zmíněný portál CODE. [15]

Před začleněním herních prvků do výuky je třeba si rozmyslet několik důležitých faktorů, ze kterých vyplyne strategie použití gamifikace:

- *Pro koho je výuka koncipována?*
- *Jaké výukové objekty budou využity?*
- *Jak bude výuka strukturována?*
- *Jaké nástroje budou mít žáci/učitelé k dispozici?*
- *Jaké herní mechanismy budou do výuky začleněny? [4]*

Gamifikace výuky ve správném provedení je vhodná pro aktivaci, upoutání pozornosti a řízené objevování žáků. [4]

2.7.4 Řízené objevování

Při použití heuristické metody žák prochází, zčásti za vedení učitele a zčásti samostatně, procesem objevování poznatku. Tento proces odráží ve zjednodušené podobě, odpovídající možnostem žáka, poznávací cyklus tak, jak probíhá ve vědě: od identifikace problému a formulace hypotéz, přes projekt výzkumu, jeho provedení a zpracování jeho výsledků, k interpretaci výsledků a vyslovení závěrů s ohledem na testovanou hypotézu. [23]

Při použití této metody se žák aktivně spolupodílí na hledání a objevování poznatků, jímž se má učit. Učitel řídí proces objevování pomocí otázek a pokynů. Obtížnost volí dle dispozic žáka. [23]

Výhodou heuristické metody vyučování je, že motivuje žáka, je zábavná, žáci se aktivně účastní výuky a mají radost z toho, že něco sami vyřeší. Vede k jasnějšímu a hlubšímu pochopení učiva v souvislostech. Brání tomu, aby učení sklouzlo do povrchního memorování. Na druhou stranu metoda klade vysoké nároky na učitele, je pracnější, učitel musí leckdy improvizovat. Metoda nevede žáky, aby se naučili naslouchat souvislému výkladu při přednášce, což je důležité ve vyšších ročnících středních škol pro přípravu na vysokoškolský způsob studia. [23]

Aplikováním heuristické metody řízeného objevování do oblasti programované, gamifikované výuky může vyučující vytvořit silně motivační a zároveň efektivní vzdělávací prostředí. [4]

3 Přehled robotických programovatelných pomůcek

V této kapitole uvedu přehled vybraných robotických programovatelných hraček, v edukačním procesu řekněme propedeutických pomůcek. Které by bylo možné využívat na střední škole, zejména v předmětech, které začínají s programováním. Většina těchto hraček má vlastní intuitivní grafické programovací prostředí, ve kterém lze začít programovat jednoduché programy ještě před tím, něž se přejde k probírání syntaxe konkrétního programovacího jazyka. Program lze jednoduše nahrát do robotické pomůcky a ta, program ihned vykoná.

Jedná se o stavebnice a robotické hračky s možností změny jejich chování díky vlastnímu programovému kódu. Žákům se dostává okamžité odezvy na jimi vytvořený program, což zvyšuje jejich motivaci. [24]

Programovatelné robotické hračky jsou ideální pro využití v začátcích programování, jelikož žáci své vytvořené programy rovnou převádějí do mechanické hračky, která vykonává jimi vytvořenou sekvenci příkazů. Tyto robotické hračky nebo stavebnice jsou díky této funkci pro žáky velmi atraktivní, je však třeba dát velký pozor na to, zda jejich nadšení plyne z novosti této pomůcky nebo zda za tímto nadšením stojí jiný přístup při tvorbě programu. [24]

Je možné obohatit například první hodinu programování s klasickým úvodním příkladem výpisu „Ahoj světe“ na obrazovku, naprogramováním robotické pomůcky, která přijede, zastaví na značce a pozdraví. Mimoděk lze přitom použít základní programové struktury, jako větvení a cykly, na které lze zpětně při probírání syntaxe konkrétního jazyka, např. C# odkázat.

Aby práce s takovouto programovatelnou robotickou pomůckou měla smysl, a každý žák si ji mohl vyzkoušet, je třeba mít jednu robotickou sadu na pracovní skupinu dvou až tří žáku. Jednou z nevýhod v českém školním prostředí je finanční a administrativní náročnost související s pořízením takovýchto pomůcek. To může zdržet nákup pomůcek, což může vést k tomu, že žáci pak nepracují s nejnovějším modelem. Proces nasazení těchto pomůcek, tedy není flexibilní a nereflektuje vývojové trendy na trhu.

Přestože trend, začleňování robotických hraček do edukačního procesu, je na vzestupu až nyní, kdy na trhu existuje několik variant robotických stavebnic a robotů vhodných pro výuku, lze podobný přístup vypozorovat již v 70. letech 20. století. [24]

Původcem těchto robotů byla mechanická želvička Logo vyvinutá Tomem Callahanem na MIT v letech 1969-1970. Tento robot byl ovládán stejnojmenným programovacím jazykem. Celá koncepce se postupem času zdokonalovala, z původního kabelového přenosu informací se přešlo na bezdrátový přenos, a došlo k designové změně podoby robota. Součástí robota byla zápisová část, která mohla zaznamenávat trasu robota, čímž se z něj stal grafický nástroj. Ve spojení s touto koncepcí vešel do povědomí pojem želví grafika. [24]

3.1 LEGO Mindstorms

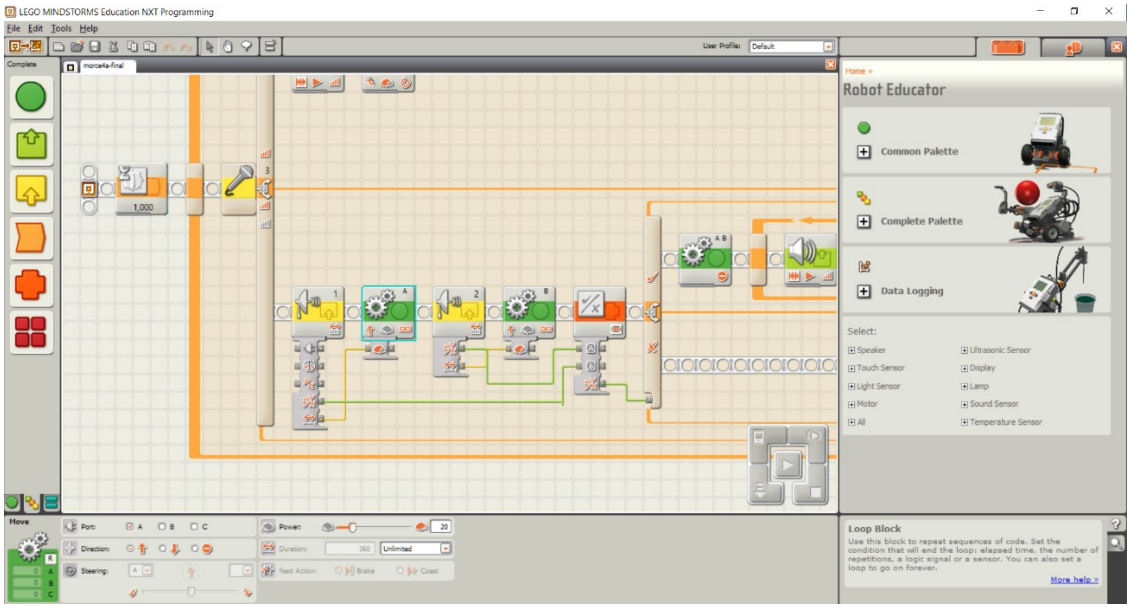
Stavebnice *LEGO® MINDSTORMS®* [25] se díky své propracovanosti stala etalonem pro ostatní stavebnice.

Základem celé stavebnice *LEGO® MINDSTORMS®* [25] je řídící jednotka tzv. *kostka*, která vyhodnocuje vstupy z jednotlivých senzoru a řídí dle programu aktívní prvky stavebnice. Do *kostky* lze zapojit kabely od jednotlivých senzorů, spínačů a servo motorů. Na *kostce* je, krom napájecí baterie, též displej s ovládacími tlačítky, kterými lze *kostku* ovládat, případně vyvolat již uložený program. Toto ovládání bez počítače je, ale nepraktické a pro výuku je praktičtější *kostku* programovat na počítači v proprietárním grafickém programovacím prostředí.

Každý blok příkazů má dodatečná nastavení, která jsou umístěna v dolní části prostředí a slouží k přesnějšímu nastavení jednotlivých senzorů. Bohužel citlivost některých senzorů je problémem, na který se musí při stavbě robota pamatovat. Jedná se o druhy fotosenzorů, které jsou ovlivňovány osvětlením okolního prostředí. [24]

Kostka má na sobě též standardní výstupky a díry, kterými ji lze mechanicky připojit k jakýmkoli prvkům stavebnice Lego. Lze tak tento minipočítač zakomponovat do jakékoliv stavby z Lego stavebnice. To může ale překvapivě býti ve výuce programování nevýhoda, neb nemalý čas výuky je třeba počítat na návrh a sestavení mechanické části robota. Tím nijak nesnižuji pěstování kreativity žáků, při návrhu a stavbě robota, či následování obrázkového algoritmu, který vede k sestavení konkrétního robota dle rozličných návodů, které jsou i součástí zmíněného programovacího prostředí. Výstavba robota může zabrat více než jednu vyučovací dvouhodinu. Pak je třeba řešit logistickou otázku s tím, že stavebnice je do další hodiny výuky blokovaná. Další hodina je obvykle za týden a nemůžou s ní pracovat další žákovské skupiny z paralelních tříd. Tuto logistiku lze vyřešit nákupem dostatečného množství stavebnic, posunem výukových plánů,

například točením skupin, nebo prostě odpadne z výuky kreativní výstavba robota a žáci programují na jíž hotových univerzálních sestavách, které jsou sestaveny jednotně tak, aby na nich šlo procvičit vše, co má učitel ve výukovém plánu.



Obrázek 8: Programovací prostředí LEGO MINDSTORMS Education NXT

Na trhu je aktuálně verze EV3, pro školní prostředí je téměř nutné pořídit verzi Education, tedy např.: *LEGO® MINDSTORMS® Education EV3 Core Set* [25], která kromě přizpůsobené výbavy, je dodávána v pevných plastových boxech s přihrádkami, kde má každá součástka své místo. Boxy lépe snesou školní zacházení a lze kontrolovat úplnost stavebnice viz. Obrázek 9.

Ve školní praxi se setkáme se staršími stavebnicemi verze NXT, patrně kvůli finanční nákladnosti. Tyto dvě verze, NXT a EV3 nejsou příliš kompatibilní, každá má dokonce jiné programovací prostředí, principiálně jsou ale shodné.

Stavebnice *LEGO® MINDSTORMS®* lze koupit i jako běžný produkt pro domácí využití. Zde je třeba apelovat na rodiče, aby zvážili investici do takovéto hračky. Domnívám se, že byť je programovací prostředí intuitivní, žák základní školy většinou není schopen bez odborného vedení zvládnout programování robota a využívá jí tak jen jako konstrukční stavebnici. Což potvrzuje i Martina Kupilíková, ředitelka Centra robotiky v Plzni a specialistka na vzdělávání učitelů, opět v článku „Co může robotický systém přinést do výuky?“ v časopise e-Mole.cz [13]:

„Na exkurzích, při kterých k nám chodí školní třídy, slýchám, že dost žáků má už nějakou dobu doma Lego MINDSTORMS a jediné, co s ním dělají, je konstrukční činnost, protože

prostě neví, jak začít s programováním. Neučí se s tím ve škole, nechodí na kroužek, rodiče tomu nerozumí, tak děti jen staví. Dost drahá stavebnice jen na konstrukci, že? … [13]

Obrázek 9: Stavebnice LEGO® MINDSTORMS® Education EV3 Core Set (zdroj: [25])

Ruku v ruce s nákupem domácí sady Lego MINDSTORMS by mělo býti angažování dítěte v nějakém kroužku zaměřeném na robotiku, nemají-li výuku s robotickými stavebnicemi na základní škole.

Cena těchto stavebnic přesahuje částku 10000Kč [25], což je podle mne hlavní překážka v jejím rozšíření. Tyto stavebnice najdeme zejména na středních školách se zaměřením na informační technologie, případně elektroniku. Avšak ani tam nepředpokládám, že budou mít k dispozici stavebnici pro každou dvojici (výukovou skupinu) žáků v ročníku. Spíše to budou jednotky kusů a v pravidelných hodinách algoritmizace, či předmětech určených přímo pro tato robotická zařízení budou probírána pouze informativně. Využití

tak spíše najdou v případě velkých školních projektů, hackathonů, v zájmových kroužcích, či v dlouhodobých maturitních pracích.

3.2 LEGO Education SPIKE Prime

Menší bratříček předchozí stavebnice je *LEGO® Education SPIKE ™ Prime.* [26] Stavebnice je podle mne daleko lépe přizpůsobena školním hodinám. Během dvouhodinové výuky lze zvládnout jak konstrukci, tak vývoj programu. Kódovací jazyk je grafický, nenáročný a na principu programovacího jazyka Scratch. Programovatelný *Hub* ve tvaru cihly je vybaven šesti vstupně výstupními porty, světelnou maticí 5x5, reproduktorem, gyroskopem, baterií a Bluetooth. [27] Už vlastně samotný *Hub* může plnit díky svému displeji nějaký smysluplný program, např. hod kostkou, či senzor otřesu. Opět sada pro školy je v plastovém boxu, kde má každá součástka místo.

K této stavebnici je opět zpracováno mnoho výukových lekcí, které jsou navrženy tak, aby žákům pomohly rozvinout abstraktní a kritické myšlení, které potřebují k řešení komplexních problémů. Lekce poskytují řadu vzdělávacích zkušeností, které se přímo vztahují k otázkám a pozorováním studentů v reálném životě, budují jejich důvěru a připravují je na život mimo školu. [26]

Obrázek 10: LEGO® Education SPIKE™ Prime Set (zdroj: [26])

Intuitivní aplikace SPIKE ™ umožňuje žákům rozvíjet možnosti kódování pomocí Scratch a Python, bez ohledu na úroveň jejich dovedností. [26] Funguje jak na počítači, tak na tabletech většiny operačních systémů. Program pro robota lze vytvářet i na tabletech, eventuelně mobilech. Čili výuku s těmito roboty lze přenést i do neinformatických předmětů a učeben, kde není k dispozici počítačové vybavení, např. do fyziky.

Díky těmto vlastnostem a ceně kolem 9000Kč si dovedu lépe představit, že škola bude schopna snáze začlenit práci s tímto robotem do běžné výuky, informatických předmětů, jak na středních školách, tak už i na druhém stupni základních škol.

3.3 Robotická stavebnice VEX

Robotické stavebnice VEX [28] jsou vhodné pro druhý stupeň základní školy a pro střední školy. Stavebnice VEX IQ [29] je s plastovými konstrukčními díly a stavebnice VEX V5 [30] je sofistikovanějším systémem s kovovými konstrukčními díly na kterou mohou žáci přejít u robustnějších řešení. Další výhodou je standardně dodávané dálkové ovládání, které umožňuje testovat mechanické vlastnosti robota a ověřit si tak funkčnost navrhovaného řešení ještě před osazením čidly a před programováním autonomního pohybu. [13]

Součástí systému jsou i konstrukční návody na sestavení různých robotů „krok za krokem“, lze ale sestavit i vlastní konstrukci. To vnímám, stejně jako u Lego MINDSTORMS, pro začlenění do pravidelné výuky jako problém, neb výstavba a otestování konstrukce robota je časově náročná záležitost, která se jen tak nevměstná do vyučovací hodiny.

Součástí stavebnice je vždy „mozek“ robota, tedy kontrolér. U VEX IQ má možnost připojit až 12 periferií, tj. čidel nebo motorů, u VEX V5 je to až 20 čidel nebo motorů, krom toho kontrolér má velký dotykový displej. Dále ve stavebnici, najdeme krom konstrukčních prvků i gyro senzor, nárazníkový senzor, ultrazvukový senzor vzdálenosti, senzor barvy aj. Pro školní provoz je stavebnice opět v plastovém boxu s přihrádkami. [13]

K programování robotů můžeme použít VEXcode Blocks [30], což je grafické prostředí s přetahováním programových bloků vycházející z prostředí Scratch, nebo můžeme použít i textový programovací nástroj VEXcode Text [30].

Obrázek 11: Programovatelný robot VEX V5 (zdroj: [30])

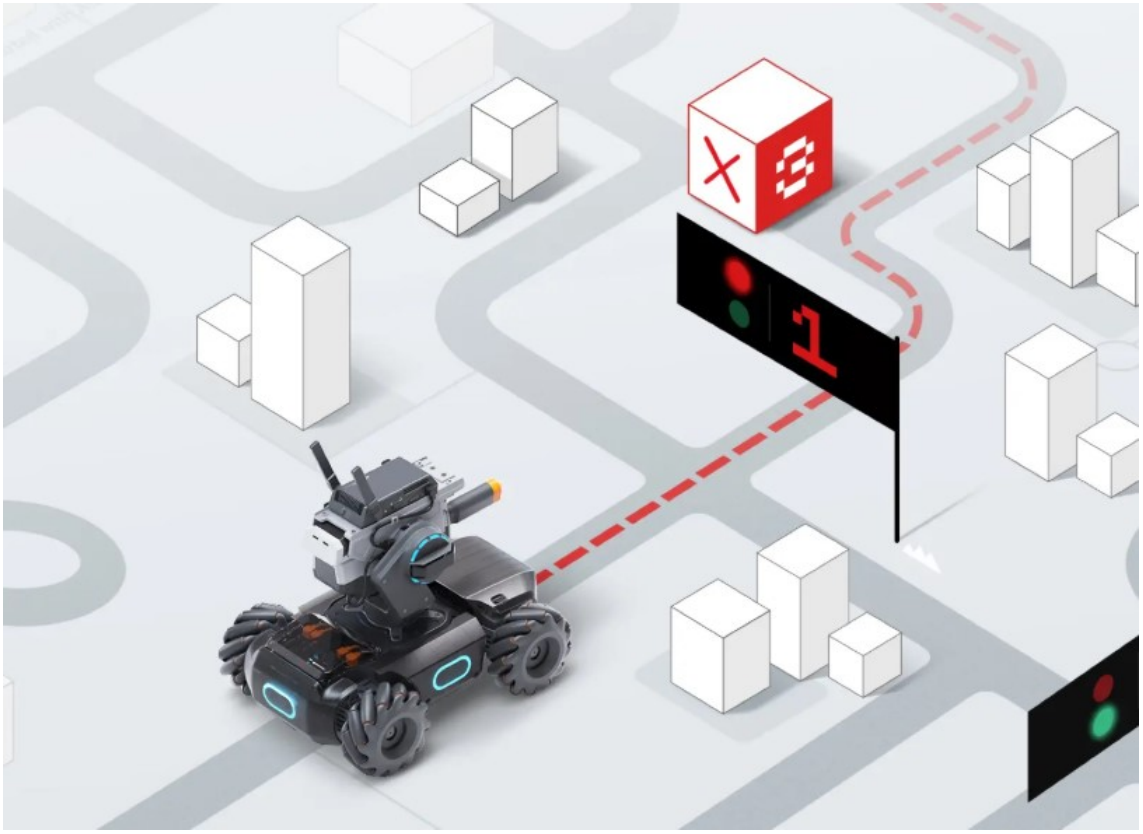
Cena stavebnic VEX IQ přesahuje částku 14000Kč [31], což je podle mne hlavní překážka v jejím rozšíření. V běžné výuce technických středních škol se zatím asi nenajdou. Využití tak spíše bude v případě velkých školních projektů, hackathonů, v zájmových kroužcích, či v dlouhodobých maturitních pracích.

Stavebnice VEX ERD V5 kit má cenu nad 72000Kč [31], čímž je používání na střední škole nereálné.

3.4 DJI RoboMaster S1

RoboMaster S1 [32] je vzdělávací herní robot vytvořený tak aby využil potenciál každého studenta. Obsahuje nejmodernější technologie, rozličné senzory, obrané i útočné prvky. Poskytne uživatelům důkladné pochopení vědy, matematiky, fyziky a programování prostřednictvím různých režimů a umělé inteligence. [32]

Jde o hračku tanku, která lze řídit dálkovým ovladačem, a to i z pohledu robota (FPV) pomocí kamery, nebo programem psaným v grafickém Scratch, nebo Pythonu. Propojuje tak digitální svět se skutečným, přivádí tak abstraktní teorie k životu. [32]



Obrázek 12: DJI RoboMaster S1 s možností autonomního řízení (zdroj: [32])

Do programu lze začlenit inteligentní funkce, jako rozpoznání čáry, značek, lidí, tlesknutí, gest a rozpoznání druhého robota. Je tedy možné rozpracovat myšlenky autonomního řízení, či se účastnit soutěží pořádaných výrobcem, či využít přidružené *RoboAcademy* [32] s videokurzy.

Vzhledem k ceně nad 12000Kč a k poněkud militantnímu účelu robota se nejspíše neuplatní při pravidelné výuce na střední škole. Nicméně myslím, že jeho potenciál je ve volnočasových aktivitách, či technických kroužcích pro studenty středních a vysokých škol.

3.5 BBC micro:bit

Idea mikropočítače micro:bit vznikla v letech 2014-2015 z iniciativy britské mediální společnosti BBC v nostalgické návaznosti na téměř čtyřicet let starý projekt stolního počítače Micro. Stejně jako projekt Micro, vznikal i micro:bit od základu jako učební pomůcka, kladoucí si za úkol zasvětit žáky základních škol, do tajů výpočetní techniky. [13]

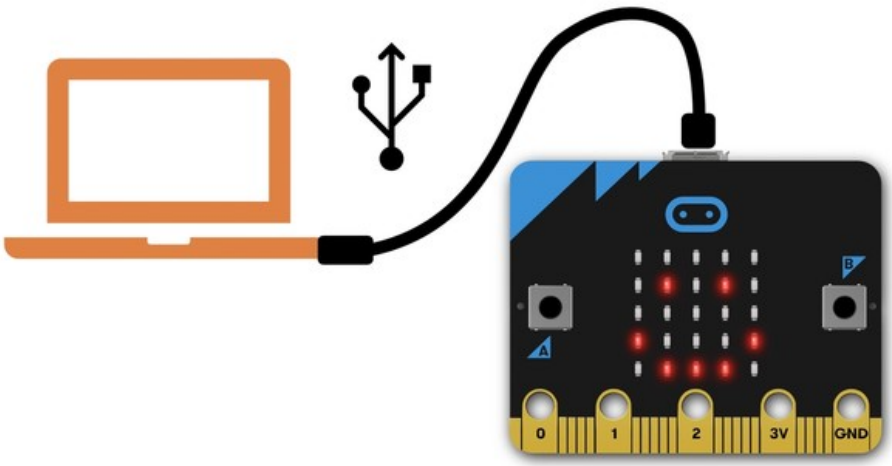
Není na škodu si připomenout, že právě vývoj počítače Micro v britské firmě Acorn Computers položil základy mikroprocesorové architektury ARM, které v současnosti vládne světu spotřební elektroniky a je použita i v micro:bitu. [13]

Hlavní motivací pro vývoj micro:bitu byla snaha odlákat mládež od pasívního používání různých elektronických hraček a mobilů a přivést ji k vlastní tvůrčí činnosti, rozvíjející její informatické myšlení. [13]

Učitelé hodnotí micro:bit vesměs kladně, neb učinil výuku zábavnější a dokázal žákům, že programování není obtížné. [13]

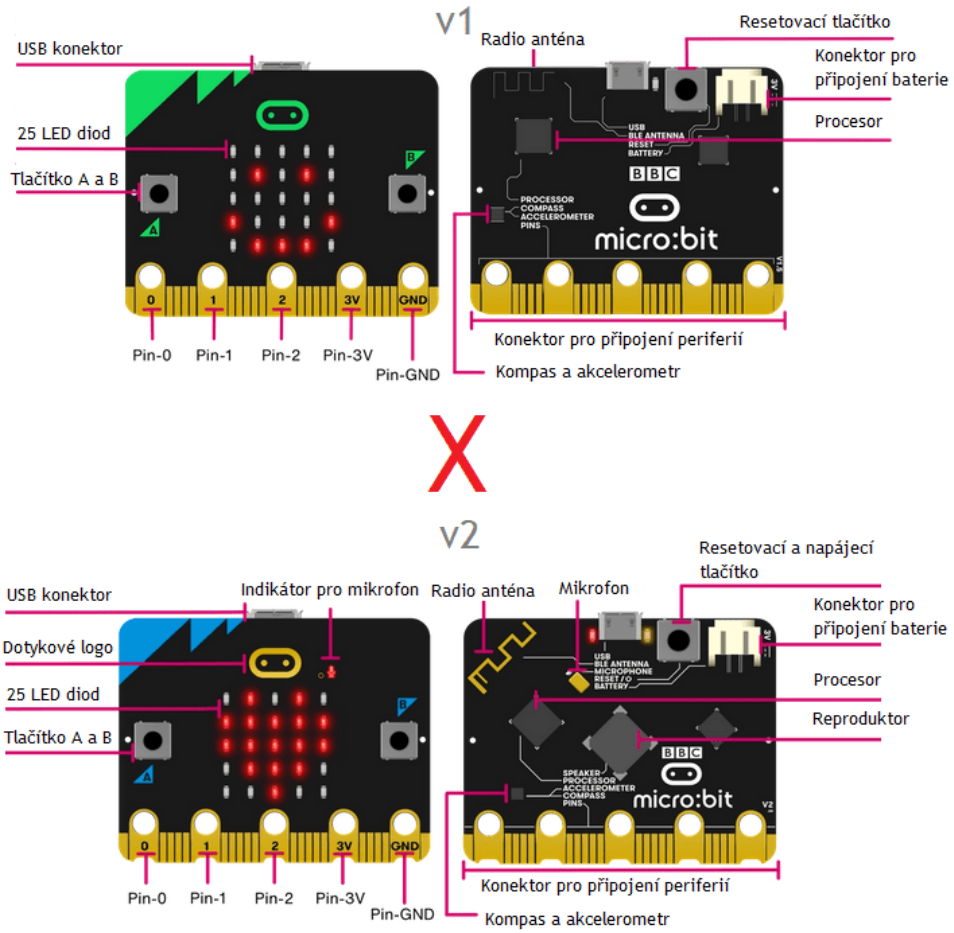
Mikropočítačem, který můžete naprogramovat v blokovém Scratchi a ti pokročilejší i v Javascriptu a Pythonu, je britský micro:bit. Prototypovací destička slavící v posledních několika letech úspěch jako skvělá učební pomůcka a je to takové Arduino pro nejmenší a začínající programátory. [33]

Micro:bit nevypadá jako hračka, ale jako elektronické zařízení, respektive základní deska plošného spoje s mikroprocesorem. Žáci si tak mohou uvědomit, co se skrývá uvnitř elektronických přístrojů. Deska plošného spoje je vyrobena kvalitně a snese školní zacházení. Přesto je vhodné vysvětlit žákům, že by jí měli držet za okraje, stejně jako např. DVD médium. Je napájena jen 3V, což je vhodné pro děti. Žáci tak operují pouze s tužkovými bateriemi. U rozsáhlejších projektů s micro:bit může být 3V logika a napájení problém. Vyrábějí se ale převodníky, ty pak umožňují připojit i neoficiální periferie, které pracují s napětím 5V.



Obrázek 13: BBC micro:bit připojený do počítače (zdroj: [34])

Samotná prototypovací destička micro:bitu o rozměrech 5x4cm krom ARM procesoru již obsahuje mnoho zajímavých prvků, jako je matice 5x5 LED diod, dvě programovatelná tlačítka, teploměr, magnetometr a akcelerometr, Bluetooth, napájecí a microUSB konektor. Aktuálně jsou k dispozici dvě verze, původní V1 a vylepšená V2, viz. Obrázek 14. Se samotnou destičkou již můžeme pracovat a vytvářet pro ni program.



Obrázek 14: Popis kapesního počítače micro:bit V1 a V2 (zdroj: [35])

Další výhodou je cena. Samotná deska stojí pouze kolem 500Kč [36]. To vede k tomu, že svůj micro:bit může mít k dispozici opravdu každý žák. Což na podzim roku 2016 udělala britská vláda a darovala micro:bit každému žáku 7. třídy základní školy ve své zemi. [13] Vhodné je dokoupit rozšiřující modul pro nepájivé kontaktní pole. Další externí diskrétní součástky, jako LED diody, rezistory tranzistory, nepájivá pole s propojovacími kablíky lze dokoupit v obchodě s elektronickými součástkami. Pro začátek patrně většina sáhne po nějaké základní, či startovací sadě, případně rovnou po sadě pro chytrou domácnost, či po sadě pro internet věcí (IoT).

Z toho plyne, že pomocí micro:bitu lze demonstrovat i základní principy Průmyslu 4.0, především inteligentní komunikaci mezi jednotlivými zařízeními. [13]

Cena většiny sad se s přehledem vejde do 2000Kč [36]. Ještě je třeba zmínit, že pro micro:bit se vyrábějí i pojízdné sady se servomotory, čímž z micro:bitu vyrobíme prapůvodní želvu LOGO. Některé podvozky mají dokonce i držák na fixu.

Obrázek 15: vozíček ring:bit V2 pro micro:bit (zdroj: [36])

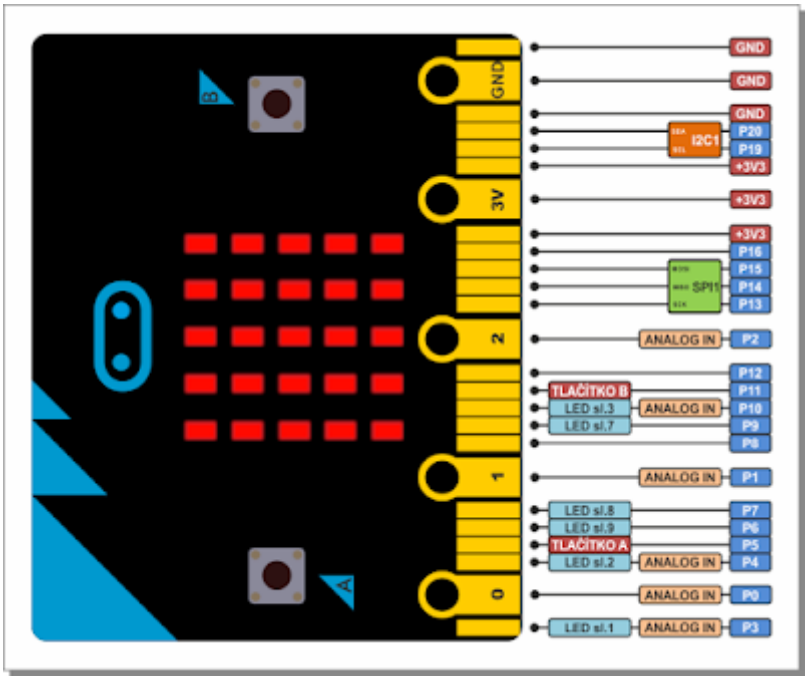
Díky všem výše zmíněným vlastnostem je micro:bit vhodný jako pomůcka k výuce algoritmizace a podpory algoritmického myšlení, kterou lze začlenit do běžné výuky, informatických předmětu, jak na středních školách, tak už i na druhém stupni základních škol.

3.5.1 Hardware micro:bitu

Základem micro:bitu je obvod SOC (System on a Chip) nRF51822[2] s 32bitovým procesorovým jádrem ARM Cortex MO a komunikačním rozhraním Bluetooth Low Energy (BLE) firmy Nordic. Díky tomuto rozhraní je možná snadná spolupráce micro:bitu s tablety nebo chytrými (smart) mobilními telefony. [13]

Mikrokontroler micro:bitu obsahuje i 10bitový A/D převodník (ADC), jehož pomocí je možno měřit při školních pokusech ty fyzikální veličiny, které jdou převést na elektrický proud. Naměřené hodnoty lze přes rozhraní USB nebo Bluetooth snadno přenášet do PC a zobrazovat je některým terminálovým programem. [13]

Všechny vstupní a výstupní signály a rozhraní jsou vyvedeny na sběrnici – zlaceným přímým rozšiřujícím konektorem na okraji desky, viz. Obrázek 16, ke kterému lze připojit další periferie.



Obrázek 16: Zapojení PINů na konektoru micro:bitu (zdroj: [37])

Vhodně je vymyšlen tvar PINů 0, 1, 2, 3V a GND, které jsou širší a mají vytvořeny otvory. Lze tak k nim bezpečně připojit kabel s krokosvorkou, či zasunout kabel zakončený kolíkem (konektorem s banánkem o průměru 4mm). Čili lze připojit různé měřící přístroje a další zařízení. Výše zmíněné otvory v pinech slouží též jako pevné a zároveň vodivé uchycení desky mikro:bitu například na nějaký podvozek, viz. Obrázek 15.

[2] Obvod SoC nRF51822 obsahuje 256kB paměti Flash a 16kB paměti SRAM. [13]

Na desce micro:bitu je jednoduchý displej, složený z matice 5 x 5 červených LED, na němž lze zobrazovat jednoduché obrázky, ikony nebo znaky, dvě uživatelsky programovatelná tlačítka, senzor zrychlení (akcelerometr) a senzor magnetického pole (kompas). Velmi důležité je pro začátečníky také tlačítko RESET, které uvede při zbloudění nepovedeného programu jediným stiskem micro:bit do výchozího stavu [13] (viz. Obrázek 14).

Toto vybavení, v součinnosti s nízkým napájecím napětím a nepatrným odběrem proudu umožňuje micro:bit přímo používat jako tzv. nositelnou („wearable") elektroniku nebo vytvářet zábavné hračky a hříčky bez nutnosti připojovat další zařízení. [13]

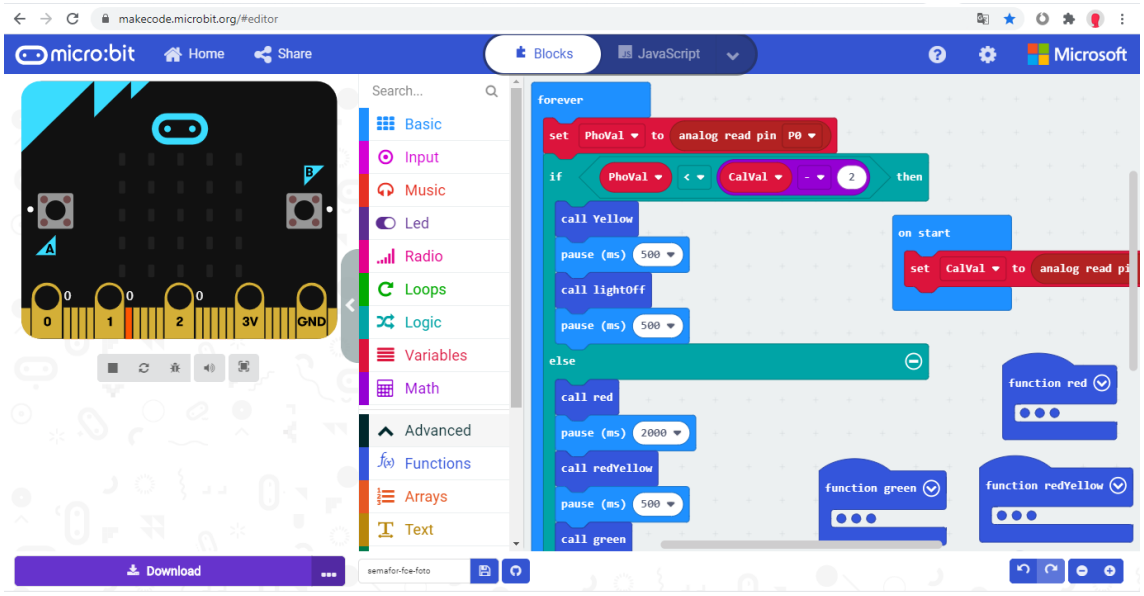
Micro:bit je otevřená platforma, čili k dispozici je kompletní programátorská i výrobní dokumentace, čili kreativitě nejsou kladeny žádné překážky. [13]

3.5.2 Software micro:bitu

Nejvhodnější a nejsnáze dostupné vývojové prostředí pro micro:bit je Microsoft *MakeCode* [34] dostupný z webové stránky na adrese https://makecode.microbit.org/. Prostředí je tak okamžitě dostupné a běží, jak v počítačích, tak v tabletech a mobilech. Programovací prostředí vychází z grafického jazyka Scratch. Programuje se pouhým přetahováním grafických bloků, které představují jednotlivé části programového kódu, z nabídky příkazů do pracovní plochy právě vytvářeného programu. Na stránce programovacího prostředí je simulátor micro:bitu, který průběžně zobrazuje chování programu. Též lze přepnout do režimu krokování.

Výsledný program se uloží přímo z webové stránky stáhnutím do souboru s příponou HEX (*.hex). Do připojeného micro:bitu přes USB se program v souboru uloží stejně jednoduše, jako na přenosný flash disk. Příslušný webový prohlížeč lze samozřejmě nastavit tak, aby programovací prostředí uložilo program přímo do micro:bitu. V tomto případě je třeba nezapomenout výsledný program uložit ještě do počítače. Jednou z mála nevýhod micro:bitu je, že program v něm uložený nelze zpětně zobrazit v programovacím prostředí.

Nespornou výhodou je, že prostředí je lokalizované do češtiny, což lze doporučit pro začínající žáky na základní škole. V prostředí ***MakeCode*** lze kromě programování s bloky přepnout do programování v jazyce JavaScript.



Obrázek 17: Programovací prostředí MaceCode (zdroj: [34])

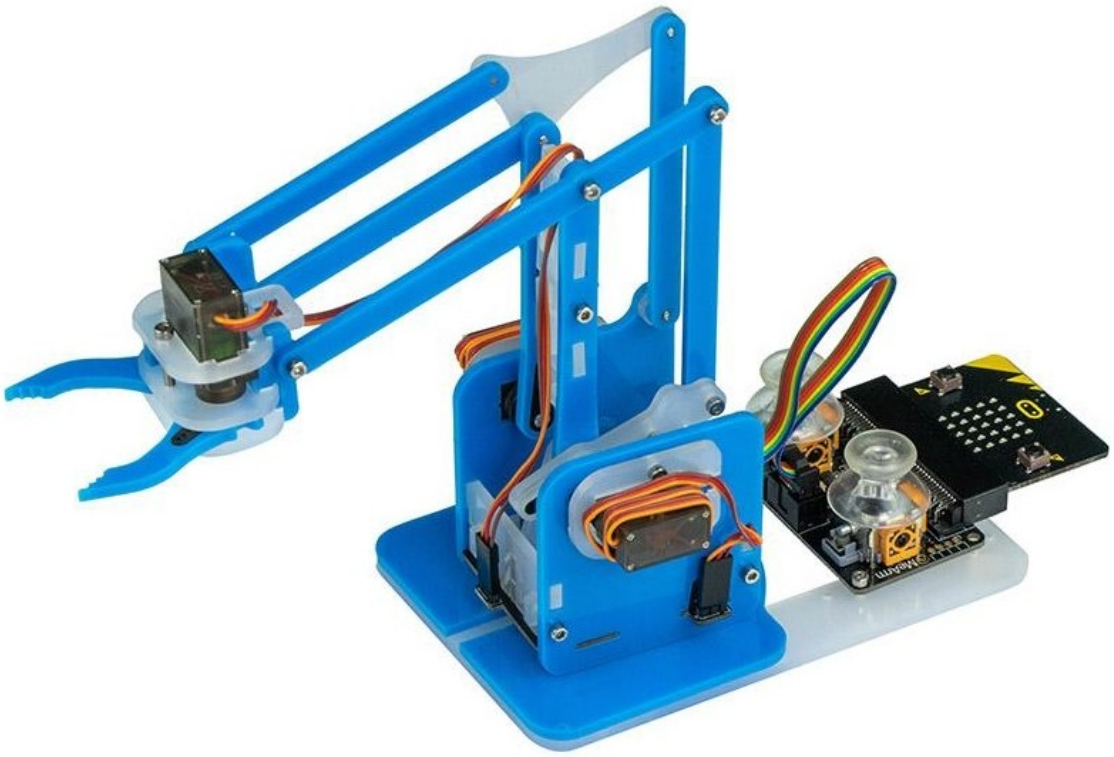
Kromě výše zmíněného editoru *existuje ještě paralelní projekt americké firmy* ***Adafruit*** *[38], jejíž vývojové prostředí je velmi podobné výše zmíněnému editoru, je s ním kompatibilní a obsahuje množství specializovaných programových bloků, určených pro snadné připojeni zařízení dodávaných k micro:bitu Adafruitem. [13]*

K dispozici jsou i vyšší programovací jazyky: ***Mu Micropython*** *[39],* ***Micropython*** *[40] a C++ pod on-line IDE* ***mbed*** *[41]. [13]*

3.5.3 Příslušenství k micro:bit

K základní desce micro:bitu lze dokoupit rozličné příslušenství od ochranných pouzder, přes přídavná relé, různá diodová pole, semafory, lampičky a další modelářské doplňky, až po robotické přípravky.

Jedna z firem zabývající se výrobou příslušenství je firma *Kitronik* [42], která nabízí různé moduly se senzory, desky pro řízení dvou stejnosměrných motorů, či celé pojízdné podvozky, různá diodová pole, reproduktory, nebo roboty, např. robotická paže *MeArm* [42], viz. Obrázek 18. Robotickou paži lze ovládat pomocí vestavěných joysticků, nebo právě programem v micro:bitu.



Obrázek 18: Robotická paže MeArm pro micro:bit (zdroj: [42])

Další firmou je *MakeKit* [43], která nabízí například přípravek *Air:Bit* s nímž v kombinaci s micro:bit postavíte létající dron, či přípravek *Hover:bit* pomocí něhož vyvinete, postavíte a naprogramujete vznášedlo, či loď.

Firma *ELECFREAKS* dodává různé rozšiřující moduly *ring:bit*, jako snímač čáry, vzdálenosti, či různé světelné moduly včetně již dříve zmíněného vozítka ring:bit, viz. Obrázek 15. Nabízí také sofistikovanou stavebnici všesměrového vozítka *Wonder Rugged Car* [44] ovládané micro:bitem.

Zajímavý je i průvodce příslušenstvím na oficiálních stránkách micro:bitu (https://microbit.org/buy/accessories/) [45], či český eshop *HW Kitchen* [36].

3.6 Arduino

Původně italský projekt z roku 2005, který si při svém vzniku nekladl velké edukativní ambice, je vyvíjen jako open-source a open-hardware. Otevřenost tohoto projektu umožnila výrobu levnějších plně kompatibilních klonů, což způsobilo oblibu Arduina, a to se tak stalo synonymem domácího bastlení. [33] Samotné Arduino, v aktuální verzi *Uno (rev 3),* je malá prototypová destička o velikosti 68x53mm, která obsahuje

mikrokontroler *Atmel*, krystal, napájení 5V, převodník pro komunikaci s počítačem a vstupní / výstupní (I/O), digitální i analogové PINy.

Každá deska má většinu I/O pinů přístupných přes standardizované patice, do kterých se jednoduše připojují další obvody, obvykle podobného formátu (Arduino *Schieldy*). Arduino deska získává údaje z různých snímačů a senzorů, např.: tlačítko, snímač osvětlení, vzdálenosti a na základě těchto údajů ovládá výstupy, např. rozsvítí LED, či světelný okruh, nebo ovládá motor.

Hlavními výhodami Arduino platformy je jednoduchost použití, obrovské množství kompatibilního hardware a Arduino shieldů, dobrá cena (kolem 650 Kč za Arduino Uno) oproti kitům a stavebnicím, ale hlavně podpora obrovské komunity Arduino nadšenců. Díky této celosvětové skupině je možné nalézt Arduino návody téměř na cokoliv, od použití Arduina jako toustovače, přes stavbu 3D tiskárny a robota až po jeho vyslání do vesmíru jako satelitu. Možnosti využití Arduino open source platformy jsou téměř nekonečné, záleží jen na vaší fantazii. [46]

Klasické Arduino předpokládá zdatnějšího programátora se znalostí C++. V programu se volají externí knihovny, obvykle vytvářené členy programátorské komunity, což ne vždy zaručuje jejich bezchybnost. Programuje se v Arduino IDE v jazyce Arduino založeném na jazyce Wiring. [46]

Arduino je vhodné a finančně dostupné pro střední školy se zaměřením na elektrotechniku. Předpokládal bych, že by pro něj měl být vyčleněn samostatný předmět s dílčím zaměřením např. na dnes moderní internet věcí (IoT).

3.7 Raspberry Pi

Britské Raspberry Pi má možná leckdo z vás díky HDMI připojené k televizoru, kde slouží jako síťový přehrávač filmů s multimediálním centrem Kodi. Maličký a levný linuxový počítač s armovým procesorem lze ale použít i pro mnohem rozsáhlejší elektrické projekty – třeba k ovládání chytré domácnosti. [33]

Jelikož se jedná o linuxový počítač, programovat v něm můžete prakticky v libovolném jazyku od Scratche (po doinstalování podpory) po C/C++, Python, Javascript aj. [33]

Raspberry Pi je malý počítač, který může dělat mnoho rozličných věcí, lze k němu připojit monitor, klávesnici a myš, lze připojit do internetu a má slot pro SD kartu.

Na prototypovací destičce nalezneme konektory USB, HDMI, MicroUSB, Ethernetový port, Audio jack, slot SD karty a I/O PINy pro připojení dalších periferií, jako LED a tlačítka. [47] Před prvotním použitím je třeba do minipočítače Raspberry Pi nainstalovat příslušnou distribuci Linuxu *Raspbian.*

Raspberry Pi má dobrý edukativní potenciál, ale ne jako doplněk k výuce programování, kde každý žák již obvykle má k dispozici plnohodnotný počítač.

4 Modelové úlohy se zapojením robotické pomůcky

Pro zapojení programovatelné robotické pomůcky do výuky byl vybrán předmět Úvod do programování. Předmět je vyučován v 1. a 2. ročníku střední školy oboru Informační technologie jednou týdně dvě spojené hodiny. Třída je dělená na poloviny. Čili skupina má maximálně 16 žáků, každý žák má k dispozici počítač a potřebné programové vybavení a přístup k informačním zdrojům.

Výuka probíhá ve dvouhodinových cvičeních v odborných učebnách. Na začátku vyučovací jednotky je pomocí frontálního výkladu a názorných ukázek vysvětlen potřebný teoretický základ. Žák potom individuálně na počítačích procvičuje získané teoretické znalosti, přitom postupuje od jednodušších úloh ke složitějším. Učitel při výkladu volí vhodné tempo výkladu a procvičování. Začínající programátory nesmí odradit příliš náročnými požadavky a musí u nich vzbudit zájem o programování. [48]

Učitel se individuálně věnuje nadaným žákům a žákům s částečnými znalostmi programování, připravuje pro ně různě náročné varianty příkladů, využívá dle možností dílčích znalostí některých žáků. [48]

Pro výuku programování je využíván i volně dostupný software a studentské licence tak, aby žák mohl tento software používat i doma při řešení domácích úkolů a rozvíjení programátorských dovedností. Žák má k dispozici potřebné výukové materiály v elektronické podobě. [48]

Ve vybraném předmětu se vyučuje programovací jazyk C# ve vývojovém prostředí *Visual studio 2019* [49]. V 1. ročníku se probírají základní programové struktury a funkce, jednoduché i strukturované datové typy na vývoji konzolových aplikací. 2. ročník je pak věnován procvičování látky probrané v 1. ročníku, po prvním čtvrtletí se přejde k programování okenních aplikací s důrazem na objektově orientované programování a na prvky předchozí látky se nabalují metody pracující se soubory a grafikou. Výuku lze členit na tři velké projekty a předepsanou dílčí látku lze probrat v rámci projektů *Textový editor* a *Grafický editor*. Výuka je zakončena větším komplexním částečně samostatným projektem typu „Kalkulačka“, resp. nadaní žáci mají možnost volby tématu. Tematické celky probírané v jednotlivých ročnících jsou vypsány v tabulce (Tabulka 1). Na konci druhého ročníku si žáci volí zaměření, tedy zda budou nadále pokračovat předmětem Programování, či si zvolí jiný profilový předmět. Bohužel si většinou žáci vybírají jiné

profilové předměty a v programování nepokračují, což by mohla zvrátit změna stylu výuky.

Tabulka 1: Rozdělení tematických celků do jednotlivých ročníků (zdroj: [48])

Předmět: Úvod do programování	
1. ročník *konzolové aplikace*	*Vývojové prostředí* *Algoritmizace a vývojové diagramy* *Příkazy, proměnné jednoduchých datových typů, operátory* *Řízení běhu programu* *Funkce* *Strukturované datové typy* *Pokročilejší koncepty v programování*
2. ročník *okenní aplikace*	*Úvod do OOP* *Programování okenních aplikací* *Víceformulářové aplikace* *Práce se soubory* *Vytváření vlastních tříd* *Grafika* *Vyhledávácí a řadící algoritmy, efektivnost algoritmů*

Z důvodů personálních a z důvodu přesně plnit ve výuce náplň tematických plánů byl vybrán pro zapojení programovatelné robotické pomůcky 2. ročník předmětu *Úvod do programování*. Předpokládané použití pro zapojení pomůcky je zejména v pasážích věnovaných opakování programových a datových struktur z prvního ročníku. Další potenciál na uvolnění potřebného počtu hodin, pro zapojení programovatelné robotické pomůcky, je předběžná shoda na vypuštění tématu *Vyhledávácí a řadící algoritmy* v následující změně ŠVP.

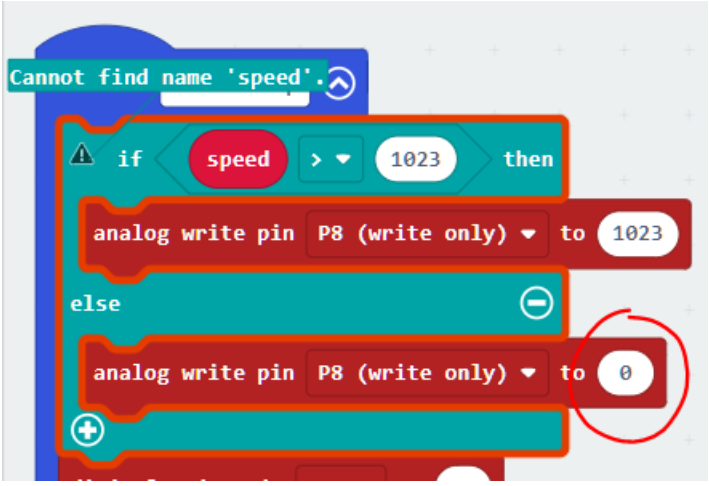
4.1 Výběr programovatelné robotické pomůcky

Paralelně, v rámci předmětu *Hardware a sítě cvičení,* mají žáci ve druhém ročníku jeden blok výuky věnovaný *Programování mikrořadiče Arduino,* [48] výuka je ale cílená na seznámení, a ne toliko na programování s Arduinem.

Pro zapojení programovatelné robotické pomůcky do stávající výuky programování byl zvolen **micro:bit**, neb byl k dispozici, dále zejména kvůli své jednoduchosti použití, které umožňuje prakticky ihned začít tvořit program, příznivé ceně, nezdvojení výuky s Arduinem ve výše zmíněném paralelním předmětu a zejména kvůli grafickému blokovému programovacímu prostředí *MakeCode*, které vychází z jazyka Scratch. Toto prostředí poskytne žákům jiný náhled a jinou perspektivu, než na kterou jsou z hodin zvyklí.

4.2 Syntaktické chyby micro:bitu a prostředí MakeCode

Grafické vývojové prostředí *MakeCode* je pokročilé prostředí, které splňuje náležitosti k eliminaci syntaktických chyb popsaných v kapitole 2.6.1. Umožňuje tedy programovat graficky pomocí bloků, a také pomocí textových příkazů psaných v jazyce JavaScript. V textovém režimu je k dispozici nápověda *IntelliSense*, v grafické režimu je k dispozici nápověda, přes pravé tlačítko myši, ke každému bloku. V textovém režimu jsou syntaktické chyby ihned podtrhávány a v dolní části prostředí vypíše podrobnější popis chyby. V grafickém prostředí nelze většinou chybně zvolený blok vůbec přichytit na zamýšlené místo. V případě, že se v blokovém režimu programátorovi přesto podaří syntaktickou chybu vytvořit, prostředí buď chybný blok zvýrazní a označí vykřičníkem, kde lze nalézt podrobnější popis chyby viz. Obrázek 19, nebo se snaží chybu opravit, např. automatickým odebráním nedefinované proměnné *speed*, což je červeně zakroužkováno na témže obrázku.



Obrázek 19: Zvýraznění syntaktických chyb v prostředí MakeCode

V případě, že se i přesto podaří nahrát příkaz do *micro:bitu*, který není schopen zpracovat, zobrazí *micro:bit* příslušné číslo chyby na displeji. Tabulky chyb lze dohledat

v dokumentaci *micro:bitu* [50]. Aktuálně typickým příkladem je, že programátor nahraje program s příkazy pro *micro:bit V2* do *micro:bitu V1*. Poté se zobrazí na displeji chybový kód „☹ *927*“.

4.3 Popis jednotlivých lekcí s použitím micro:bit

Na začátku první hodiny je nutné žáky seznámit s micro:bitem a s celou stavebnicí *BBC micro:bit Starter kit.* [36] Je třeba počítat více času na teoretický úvod a případnou administrativu s rozdáním stavebnic.

Následující lekce jsou koncipovány tak, aby mohly být vloženy do úvodních hodin opakování látky předchozího ročníku předmětu *Úvod do programování*. Povzbudili tak u žáků kreativitu, poskytli jim jiný úhel pohledu na dané téma a motivovali je k další práci.

4.3.1 Lekce 1 – semafor pro chodce

Žáci jsou seznámeni se schématem micro:bitu dle Obrázek 14 *Popis kapesního počítače micro:bit (zdroj:* [37]*)*. pomocí kterého lze žákům popsat že micro:bit obsahuje procesor, konektor na baterii i Micro USB, tlačítka „A“ a „B“, displej z červených LED, magnetometr, tedy kompas a akcelerometr, čili tzv. gyroskop, podle nějž micro:bit pozná náklon, či zatřesení, Bluetooth na propojení s počítačem, či tabletem, nebo dokonce k propojení dvou micro:bitů navzájem, čímž lze ovládat jeden micro:bit druhým. Je třeba též zmínit zlacený přímý rozšiřující konektor na okraji desky, ke kterému lze připojit další periferie. Jsou pbrány jednotlivé PINy dle Obrázek 16. Je třeba žáky upozornit na opatrnou manipulaci s destičkou micro:bitu, zejména jej držet za okraje a zbytečně se nedotýkat pozlacených PINů rozšiřujícího konektoru, neb by mohlo dojít poškození micro:bitu.

Po rozdání stavebnic se žáci seznámí s obsahem příslušné micro:bitové sady (součástky, kabely, konektory). Zejména je třeba vysvětlit a pomoci žákům připojit micro:bit k nepájivému kontaktnímu poli pomocí rozšiřujícího konektoru pro kontaktní pole.

Pro první hodinu byl vybrán příklad semaforu. Zapojení je vcelku jednoduché a vše potřebné startovací sada s micro:bitem obsahuje. Pilotní výuka proběhla na Střední průmyslové škole elektrotechnické, Praha 10, V Úžlabině 320, kde se žáci i oboru Informační technologie běžně setkávají s diskrétními součástkami. Také ve volbě tématu

se semaforem je možno spatřovat jistý potenciál případné mezioborové spolupráce při různých středoškolských soutěžích a hackathonech, např. při tvorbě „chytré křižovatky“. Pro případ, že by práce s diskrétními součástkami nevyhovovala, lze dokoupit příslušenství v podobě již hotového semaforu od firmy *Kitronik* [42].

Nyní je možno žákům předložit, nebo promítnout připravený *Pracovní list lekce 1 – Semafor pro chodce* viz. Příloha 2 . Je vhodné, aby pracovní list měli žáci k dispozici elektronicky, jsou v něm odkazy, které si mohou rozkliknout. Žáci si sestaví obvod podle schématu v pracovním listu tím, že zapojí dvě LED diody s předřadnými rezistory a pomocí kablíků je propojí s PIN P0 a P1 na kontaktním poli. Nyní má každý žák, případně dvojice, připraven „semafor pro chodce“.

Nyní je vhodné přistoupit k programování micro:bitu v prostředí *MakeCode* na adrese https://makecode.microbit.org/ [34]. Je vhodné vysvětlit, žákům, jak programové prostředí přepnout z češtiny do angličtiny. V pracovním listu je první programový kód kompletně zobrazen v blokovém programovacím jazyce *MakeCode*, případně je ke stažení. V programu používáme příkazy *digital write pin* a *pause.* Pauza pouze simuluje dobu svitu konkrétní barvy na semaforu a je oproti reálnému stavu zkrácena. Učitel by měl žákům vysvětlit podstatu fungování programového bloku *on start* a *forever*, což je cyklus, který se opakuje do nekonečna[3]. Učitel upozorní žáky, že je možné během programování sledovat střídající se stavy jednotlivých PINů micro:bitu na simulaci přímo v programovém prostředí *MaceCode*. Po naprogramování aplikace učitel ukáže, jak nahrát program do zařízení micro:bit. Upozorní, že je třeba program uložit ještě do počítače, neb z micro:bit nelze program získat zpět. Zapojení micro:bitu s LED diodami by mělo střídavě blikat, obdobně jako semafor pro chodce.

Účelem této lekce bylo seznámit žáky se zařízením micro:bit, vytvořit jednoduchý program a nahrát jej do zařízení. Pro korespondenci se základním předmětem *Úvod do programování* a potřeby zopakovat látku předchozího ročníku, je třeba připomenou, že program pro micro:bit reprezentoval posloupnost příkazů – **sekvenci** a připomenout analogické programy v C# probrané v 1. ročníku, zejména pak připomenout příkaz *Console.WriteLine();*.

[3] Zde můžou mít zvídaví žáci prostor pro dotaz ohledně nekonečnosti algoritmu, jenž odporuje probrané definici popsané v kapitole 2.2 Algoritmus. Jde o výjimku a program algoritmem je.

4.3.2 Lekce 2 – automobilový semafor

Ve druhé lekci, viz. Příloha 3 *Pracovní list lekce 2 – automobilový semafor,* již žáci práci s programováním micro:bitu zvládají. Pochopili-li správně zapojení PINů micro:bitu, měli by být schopni sami rozšířit zapojení na tříbarevný semafor. Učitel pouze kontroluje zapojení, připomene nutnost polarity LED, sleduje zda žáci logicky zaměnili zelenou LED za žlutou a zelenou připojili na PIN P02. Při úpravě kódu si mají žáci osvojit práci s bloky v prostředí *MakeCode*, zejména duplikaci bloků. V případě, že některý žák je pomalejší, může si předložený kód stáhnout odkazem z pracovního listu.

V programu je záměrná chyba, semafor má jen tři stavy. Reálný semafor má však za běžného fungování stavy čtyři. Stav, kdy svítí červená současně se žlutou v algoritmu chybí. Je pravděpodobné, že to některý žák ihned sdělí, nebo to bude jasné ze simulace, či z reálného chování micro:bit a blikání LED. Je vhodné i přes odhalení chyby pustit projekci reálného semaforu na plátno (v pracovním listu je příslušný odkaz na video).

Žáci mají objevenou chybu v algoritmu samostatně opravit. V tento moment odlaďování je vhodné ukázat správné schéma zapojení LED. V případě, že už má algoritmus opravena většina žáků, je vhodné, ukázat třídě jednotlivá řešení žáků.

V případě, že budeme žáky motivovat, aby byli co nejrychlejší, patrně v programu nebudou v jednotlivých krocích (stavech semaforu) vypínat ty barvy, které již vypnuté jsou. Je vhodné, aby učitel na toto možné vypínání všech nepotřebných barev upozornil a snažil se na toto téma vyvolat diskuzi. Diskuzi je možné nechat otevřenou, jde o to, aby si žáci zkusili vytvořit vlastní názor. Nicméně, bez zbytečného vypínání LED je program jednodušší, potenciální problém zatím vidět nemusí[4].

V této lekci si měli žáci dostavět vlastní hmatatelné zařízení, což by je mohlo motivovat k další činnosti. Měli najít sémantickou chybu a opravit programový kód. Případné pokusy schopnějších žáků kombinovat projekty semaforu pro chodce a automobily je vhodné podpořit. V návaznosti na potřebu opakovat látku předchozí výuky C# je možné zopakovat možnosti ladění ve vývojovém prostředí C# a zopakovat příslušné klávesové zkratky. Z teorie je vhodné zopakovat již probrané druhy chyb algoritmu zmíněné v kapitole 2.6.1.

[4] Téma je možné opět otevřít v momentě, kdy kód semaforu bude rozšiřován o funkce, či dokonce bude reagovat na různé události denního a nočního režimu, či náhlého průjezdu vozidel integrovaného záchranného systému

Tyto dvě lekce s micro:bitem by měli být zvládnutelné v jednom dvouhodinovém cvičení. Záleží však na důkladnosti opakování a upevňování látky předchozích ročníků.

4.3.3 Lekce 3 – větvení a funkce

V této lekci je třeba nejprve zopakovat definici a podstatu **funkcí:** „Potřebujeme-li opakovaně vyvolat část kódu, můžeme z této části vytvořit funkci a tu následně volat. Zpřehlední se tak i program.“ Což koresponduje s opakovanou látkou, kterou lze připomenout i procvičenými příklady s funkcemi v C# z loňského roku.

Každý žák by měl pokračovat na svém micro:bitovém projektu automobilového semaforu z předchozí lekce 2. Příloha 4 obsahuje pracovní list L*ekce 3 – větvení a funkce,* ve kterém je na úvod ukázáno, jak se tvoří a volá funkce v programovém prostředí *MakeCode*. Učitel ukáže žákům tvorbu funkce pro rozsvícení červené LED. Tvorbu funkcí pro ostatní barvy nechá na samostatné práci žáků. Učitel sleduje tvorbu žáků, zejména jak řeší funkcí stav, kdy svítí na semaforu červená se žlutou. V pracovním listu je sice naznačeno řešení funkcí *redYellow* případná jiná řešení lze však korigovat, či podpořit. Dále sleduje, zda žáci učinili nějaký vlastní závěr z debaty o vypínání všech LED z minulé lekce. Po dokončení převodu příkazů ovládající rozsvěcení jednotlivých barev do funkcí by se chování semaforu nemělo změnit. U takto jednoduchých programu nedochází k zjednodušení a zpřehlednění programového kódu. Žádnou funkci také zatím nevoláme opakovaně.

Po tomto příkladu si žáci zopakovali funkce a naučili se s nimi pracovat v prostředí *MakeCode*. Upravili program pro micro:bit a připravili jej tak na budoucí rozšíření.

Další úkol v lekci 3 je na procvičení větvení. Využijeme toho, že semafor se např. na noc přepíná do režimu, kdy bliká pouze žlutá barva. Pro jednoduchost budou zatím žáci ovládat přepínání pomocí tlačítka „A“ na micro:bitu. Držím-li tlačítko „A“, začne semafor blikat žlutě[5], pustím-li tlačítko, vrátí se do normálního již naprogramovaného režimu. Je třeba žákům vysvětlit, kde najdou v *MakeCode* blok větvení a podmínky, čímž žákům připomeneme nutnost podmínky, podle které se rozhoduje, která větev příkazů větvení se provede. Vhodné je připomenou žákům, že existuje úplné a neúplné větvení a

[5] V pracovním listu je odkaz na video ukázku, jak by šlo řešit přepínání pomocí fotorezistoru, který identifikuje den a noc. Zvídavé žáky je možné, ve snaze zprovoznit tento senzor, podpořit.

jejich využití. Žáci mají výše zmíněné rozšíření naprogramovat sami. Měli by opakovaně volat funkci pro rozsvícení žluté barvy a měli by vytvořit funkci na zhasnutí všech barev.

Tímto úkolem si žáci mají zopakovat větvení, tedy proces rozhodování programu a zhmotnit jej v podobě funkčního modelu semaforu s micro:bitem.

Tato lekce s micro:bitem by měla mít časovou náročnost jedné vyučovací hodiny. Záleží však na důkladnosti opakování a upevňování látky a korelaci s hlavním vyučovacím jazykem C#.

4.3.4 Další možné využití ve výuce

Mají-li žáci sestaven a funkční semafor dle předchozího úkolu z lekce 3. bylo by vhodné přivést žáky k prozkoumání chování semaforu. Zejména, že semafor se nepřepne do nočního režimu ihned po stisku tlačítka, ale vždy až když dokončí celý denní cyklus. To zřejmě nevadí, tak je program navržen. Je vhodné nastolit otázku, jak by šlo programově vyřešit, aby se semafor uměl v jakémkoli svém stavu okamžitě dostat do stavu červená. A v tomto stavu setrvat po dobu průjezdu vozidel integrovaného záchranného systému, což by šlo na micro:bitu simulovat tlačítkem „B“. Případnou debatu je možné směrovat k závěru, že již naprogramované řešení přepínání denního a nočního režimu není optimální pro okamžitý průjezd záchranných vozidel. A že tuto funkcionalitu lze jednoduše vyřešit programem řízeným událostmi (event-driven). Tyto událostí micro:bit umí aplikovat také. Aplikace řízené událostmi jsou též na programu běžné výuky v C# předmětu *Úvod do programování* hned v kapitole *Programování okenních aplikací,* kdy se začíná s programováním nějaké funkcionality při stisku tlačítka uživatelem. Zde je opět možné zařadit práci s micro:bitem a pro lepší pochopení a zvýšení motivace žáku je možné projekt semaforu rozšířit.

Obrázek 20: Příklad reakce micro:bitu na událost stisku tlačítka (použit: [34])

Další možné příklady s micro:bitem v kombinaci s programováním v C# již nejsou součástí této práce.

4.3.5 Lekce 4 – opravte chybu programu

V této lekci se žáci seznámí s programovatelným vozíčkem *ring:bit*, viz. Obrázek 15. Místo klasických motorů má servomotory, které se mohou kontinuálně otáčet o 360°. Lze k němu přimontovat fixu, tak aby mohl kreslit stopu. Dodávané servomotory nejsou nijak silné, přimontovaná fixa, podklad i stav baterií, výrazně ovlivňují jízdu. Pro ovládání *ring:bit* je třeba nahrát rozšíření *ringbitCar*. Bylo zjištěno, že pro jízdu na přesnost nejsou vhodné příkazové bloky *go*. Je vhodné se s kreslícím *ring.bitem* pohybovat pomaleji, aby nedocházelo k překmitům při rozjezdu a dojezdu. A je dobré korigovat rychlost levého a pravého motoru zvlášť příkazem *set*, tak aby *ring:bit* jel rovně. Délka dojezdu se pak tedy řídí rychlostí a délkou pauzy před příkazem zabrzdění.

Žáci mají opravit algoritmus chybně naprogramovaného robota „čišníka“, který nedojede k určenému stolu, což je podrobně popsáno v kapitole 2.6.2. Pro lepší názornost je příklad převeden do reality v podobě robota *ring:bita*. Který po počátečním zprovoznění umožňuje modifikovat zadání.

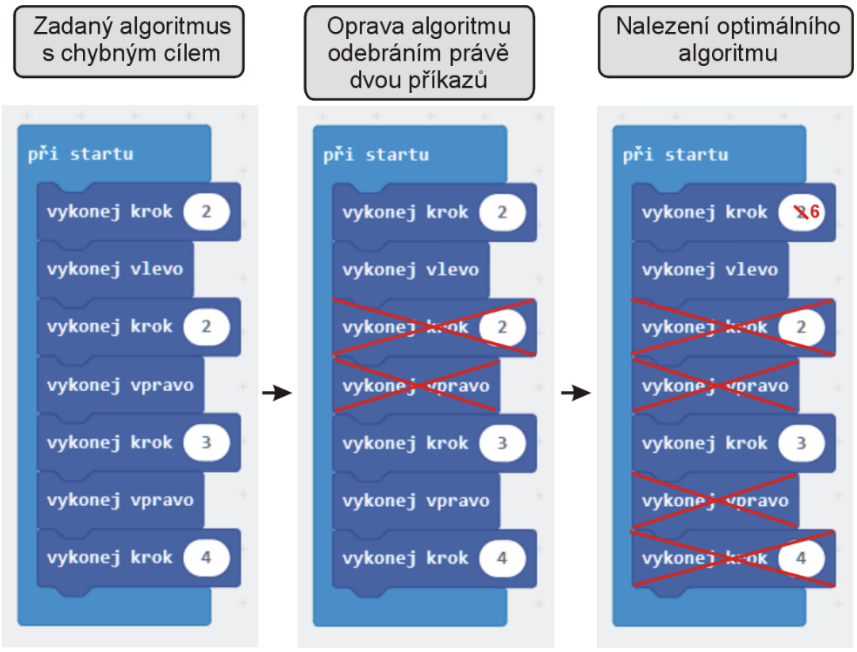
Žáci postupují dle předpřipraveného Pracovního listu lekce 4 – opravte chybu v programu, viz. Příloha 5. Zopakují si látku funkcí z předchozí lekce. Krom zadané opravy programu se musí vypořádat s korekcemi robotické soustavy, tak aby robot jezdil rovně a zatáčel přesně o 90° doleva a doprava[6].

Ukázka řešení pro případné zadání papírovou formou je popsáno v kapitole 2.6.2 Obrázek 6. V případě robotické pomůcky je příklad doplněn o následnou optimalizaci algoritmu, viz. Obrázek 21: Ukázka opravy a optimalizace algoritmu v prostředí MakeCode. Ukázka realizace programu robotem *ring:bit* je ve fotodokumentaci, viz. Příloha 7.

V tomto příkladě, s využitím metody řízeného objevování, měli žáci vyhledat sémantickou chybu přesně zadaným způsobem ji opravit a dále algoritmus optimalizovat. Dále byly zopakovány funkce, kterými je programována cesta *ring:bita*.

Tato lekce s *ring:bitem* by měla mít časovou náročnost vyučovací dvouhodiny. Časově náročné je zejména odladění nesprávných jízdních vlastností *ring:bita*. V případě, že si žáci budou model sestavovat sami, je třeba počítat další hodinu.

[6] Jde o korekci chyby hardware ring:bita vlivem nestejných vlastností levého a pravého servomotoru. Též řeší problémy s dynamikou systému při kreslení



Obrázek 21: Ukázka opravy a optimalizace algoritmu v prostředí MakeCode.

4.4 Projekt robotického auta s důrazem na práci s chybou

Vzhledem k elektrotechnickému zaměření předmětné školy byl vybrán projekt „Robotického modelu auta“ s možností dálkového ovládání. Tato myšlenka vychází ze stavebnice vozítka *ring:bit* viz. Obrázek 15, či komplexnější stavebnice *motor:bit Smart Car kit* [51] jehož rozšiřující deska *motor:bit* má k dispozici více výstupů, a tak je možno implementovat více senzorů pro sledování čáry, či vzdálenosti. Nicméně tyto stavebnicové platformy sloužily spíše pro testování. Pro zvýšení motivace žáků bylo cílem vestavět *micro:bit* do reálného RC modelu auta místo jeho řídící desky a originální ovladač nahradit druhým *micro:bitem* spárovaným přes Bluetooth. Kromě dálkového ovládání měl mít model auta nějaký naprogramovaný autonomní prvek, např. se samostatně pohybovat a před překážkou zastavit, či sledovat čáru, nebo následovat objekt před sebou, či jen provést nějaký trik. Žáci mohou buď znovu oživit nějaké vhodné vlastní rozbité RC auto. Případně lze využít možnost některých e-shopů, které odprodávají nefunkční reklamované modely za zlomek ceny, ideálně s vadnou řídící deskou. Pro modelový projekt byl vybrán červený vodotěsný cruiser (např: https://www.rcobchod.cz/vyhledavani/?sWord=Vodotěsný+CRUISER) ve velikosti 1/10, který má dostatečně velký prostor pro elektroniku a lze jich objednat více kusů. Též byl vybrán proto, že má podobné řízení, jako zmíněné vozítka *ring:bit* a *motor:bit Smart Car kit,* tedy nezávisle ovládaný levý a pravý motor, čímž lze zatáčet obdobně jako u tanku.

Úvodní část projektu navržená se stavebnicí vozítka *motor:bit Smart Car kit* lze začlenit do běžné výuky programování. Následná realizace vlastního robotického modelu auta je navržena pro školní Hackaton žákovských (mezioborových) týmů.

4.4.1 Korekce nestejných parametrů motorů

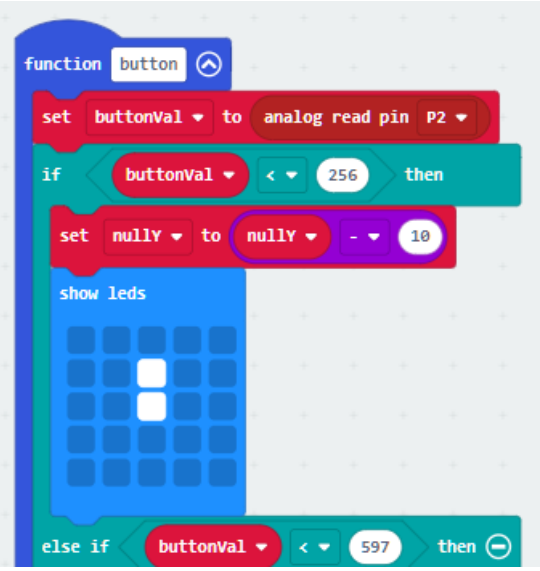
V případě, že model vozítka má dva nezávislé motory pro levé a pravé kolo, nelze očekávat, že oba motory se budou chovat stejně a že vozítko pojede rovně. Též je nutno brát v potaz, že analogový ovladač, v modelovém projektu *Joystick:bit*, bude mít přesně symetrické výchylky. Některé dálkově ovládané modely mají pro tuto korekci na ovladači tzv. trimy. Navíc charakteristika motorů může být ovlivněna i kondicí baterie, či povrchem po kterém jede. Žákům je předložen základní program na ovládání *motor:bit Smart Car* pomocí *Joystick:bit*. Mají navrhnut korekci programu tak, aby vozítko jelo rovně, ideálně aby se dalo nastavení korigovat za provozu. Dále korekci tak, aby vozítko stálo, když je ovladač nulové poloze. Potřebné programy jsou publikovány v pracovním listě Lekce 5 - dálkově řízené vozítko, viz. Příloha 6. Po nahrání programu do vozítka a ovladače je program funkční[7].

Žáci si mají vyzkoušet, že lze vozítko ovládat a zda opravdu mírně nezatáčí a případně, zda se mu v klidové poloze pomaličku netočí kola.

Mají identifikovat sémantickou chybu a počáteční hodnoty *nullX* a *nullY* nastavit na hodnotu blízkou 512 (půlky rozsahu výchylky) a nikoli 400, jak je chybně uvedeno v programu. Toto zjištění by mělo žáky nasměrovat na řešení další úlohy, a to, že by mohli změnou hodnot *nullX* a *nullY* za běhu programu po aktivaci tlačítka korigovat mírné zatáčení, či klidovou polohu vozítka[8].

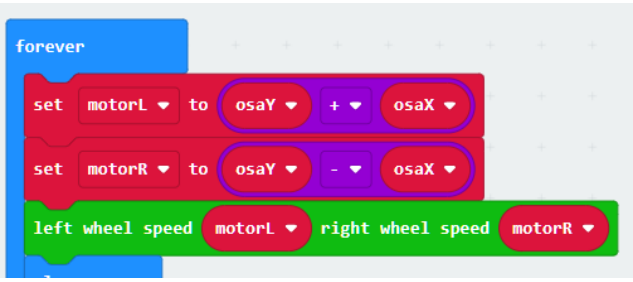
Možná oprava hodnoty *nullY* za běhu programu je patrná z Obrázek 22, kdy po stisku tlačítka odečítáme, resp. přičítáme jistou hodnotu k *nullY*, dle uvážení, např. 10.

[7] V případě více sestav ovladač-vozítko ve třídě, je třeba přidělit žákům různá čísla *radio set group*, tak aby nedocházelo k ovládání cizího vozítka.
[8] Tento postup výuky je příkladem metody řízeného objevování uvedené v kapitole 2.7.4.



Obrázek 22: Korekce programu ovladače pro jízdu vpřed

V neposlední řadě mají žáci dovodit, jaký má mít výchylka X joysticku vliv na jízdu vozítka. Opět je možné vyjít z úvodní chyby v programu, kdy žáci měli identifikovat chybně zadané výchozí hodnoty *nullX* a *nullY* v programu. Display *micro:bitu* též ukazoval jízdu mírně vpravo. Výchylka *X* se pouze prostě přičítá, či odečítá od *Y* výchylky pro levý, či pravý motor, viz. Obrázek 23.



Obrázek 23: Oprava programu pro jízdu vlevo a vpravo

V této lekci měli žáci nalézt chybu na základě chybného chování robota, který měl stát, ale jel dopředu. Na základě pochopení této chyby měli žáci doprogramovat (trimy) korekci chování celé robotické soustavy. Následně doprogramovat plynulé ovládání robota pomocí joysticku. Při programování si žáci opakují a prohlubují znalosti látky z předchozích hodin, zejména cykly, větvení a události.

Tato lekce se *Smart Car* by měla mít časovou náročnost vyučovací dvouhodiny. V případě, že si žáci budou model sestavovat sami, je třeba počítat další hodinu.

4.4.2 Hardwarová chyba dálkového ovladače

Pro dálkový ovladač robotického modelu auta byl použit *Joystick:bit V1* [52]. Jde o rozšiřující desku *micro:*bitu, která krom joysticku má k dispozici šest tlačítek, jenž lze využít k naprogramování různých efektů, např. přepínání autonomního a RC módu modelu auta, zatroubení, rozsvícení světel, či aktivaci nějakého „triku“ auta.

Na stránkách výrobce je publikován základní program viz. Obrázek 24. Již z programového kódu je patrno, že šestice tlačítek je zapojena na analogový vstup P2 *micro:bitu* a rozlišení konkrétního zmáčknutého tlačítka je realizováno tak, že stisknuté tlačítko přemostí příslušný rezistor odporového děliče.

Obrázek 24: Programový kód Joystick:bit V1 (zdroj: ELECFREAKS [52])

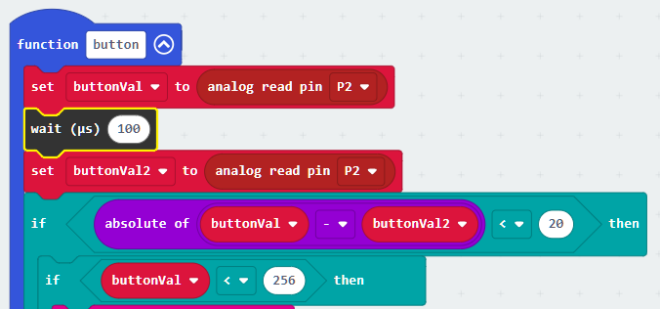
Empiricky bylo zjištěno, že v případě, že do *Joystick:bitu V1* je zapojen *micro:bit V2*, což se v praxi může stát, buď omylem, nebo shodné verze zařízení nejsou prostě k dispozici. A zároveň je aplikován přenos dat mezi *micro:bity* pomocí Bluetooth dojde v reálné

implementaci k tomu, že se chybně aktivují funkce tlačítka, aniž by bylo tlačítko stisknuto. Model auta si pak zdánlivě dělá, co chce. Zde je nutno žákům vysvětlit:

- Nutnost dodržet kompatibilitu zařízení.
- Problematiku měření analogových hodnot vstupu *micro:bitu*, vzorkování, ale také reálnou možnost tzv. přeslechu mezi součástkami.
- Problém lze vyřešit programově, tak aby se soustava HW chovala dle požadavku.

Žáci mají za úkol vyřešit problém formou opravy programu. Měli by dospět k tomu, že je třeba zkontrolovat analogovou hodnotu dalším měřením, tedy opakovaným načtením. Což je princip zcela běžný v reálném životě, např. při kontrole písemné práce před odevzdáním.

Jedno z možných řešení, které řeší výše popsaný problém je, že načteme analogovou hodnotu opakovaně s časovým odstupem. Hodnotu akceptujeme pouze pokud je rozdíl mezi první a druhou hodnotou minimální, viz. Obrázek 25[9].



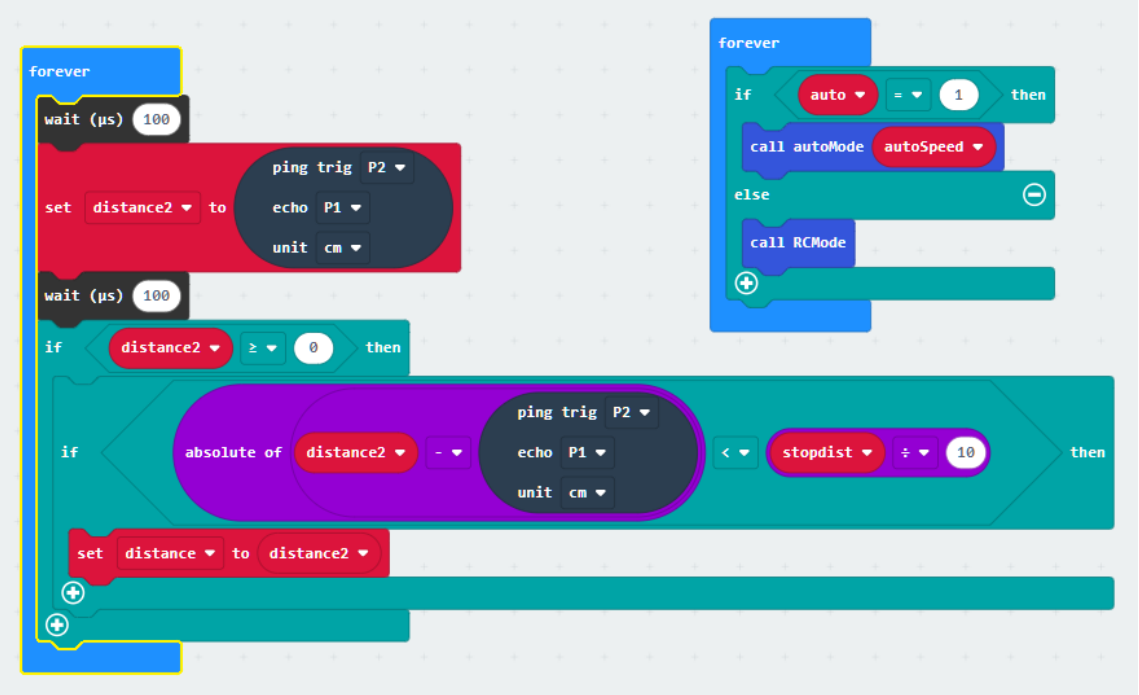
Obrázek 25: Vyřešení chybně načtené hodnoty opakovaným načtením a kontrolou

4.4.3 Chyby měření sonaru

V projektu robotického modelu auta, které umí i plavat ve vodě vznikla potřeba, aby ultrazvukový detektor vzdálenosti byl vodotěsný. Standardní s*onar:bit* tento požadavek nesplňuje. Byl tedy použit *Vodotěsný ultrazvukový měřič vzdálenosti JSN-SR04T*. Díky rozšiřující knihovně *Sonar* s tímto měřičem lze s *micro:bitem* v prostředí *MakeCode* pracovat. Měření však vykazuje příliš mnoho chyb. Pro rychle jedoucí model auta bylo třeba vyloučit chybná měření opakovaným měřením a kontrolou dvou po sobě jdoucích měření viz. Obrázek 26. Algoritmus nevyhodnocuje shodnost naměřených hodnot, ale jistou toleranci za pomocí funkce absolutní hodnoty. Princip algoritmu je stejný jako v předchozí kapitole 4.4.2, žáci by na něj měli přijít sami v okamžiku, kdy model auta

[9] Zařízení Joystick:bit V2 již nemá šest tlačítek připojených analogově k P2, ale čtyři tlačítka, digitálně propojená k P12-P15. [52]

chybně a často vyhodnocuje překážku. Navíc je měření prováděno v samostatném cyklu *forever* což u *micro:bitu* znamená, že cyklus běží paralelně ve vlastním vlákně. Měření tak běží kontinuálně[10] bez ohledu na to, co se děje v cyklu pro ovládání modelu auta.

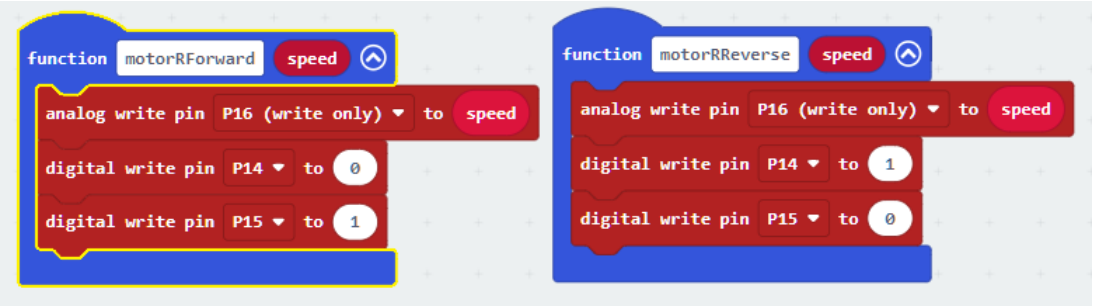


Obrázek 26: Vyřešení chybně změřené vzdálenosti opakovaným měřením a kontrolou

4.4.4 Další případy práce s chybou

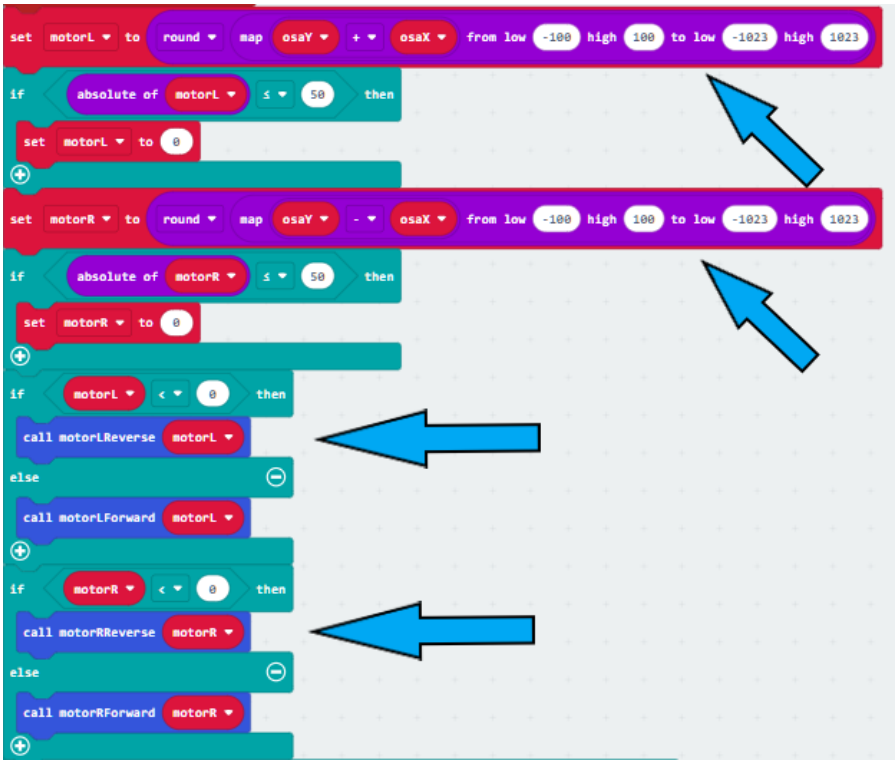
Při realizaci projektu robotického modelu auta byl místo příslušenství *motor:bit* použit *Motor Driver* [53]. Destička příslušenství je menší a výkonnější. Nemá ale kvalitní a schválené rozšíření do programovacího prostředí *MakeCode*. I s ohledem na plánované použití ještě výkonnějších H-můstku pro ovládání motorů, byly po studium ovládacích knihoven *Motor Driveru* výrobce Waveshare [53] naprogramovány vlastní funkce viz. Obrázek 27. Každý motor potřebuje dva digitální výstupy *micro:bitu*, kterými se řídí směr otáčení, zastavení a volnoběh. A analogový výstup, kterým se řídí rychlost.

[10] Ultrazvukový měřič vzdálenosti JSN-SR04T lze přepájet do režimu 3 ve kterém probíhá měření kontinuálně a data se vyhodnocují sériově (UART). Což by mohlo být pro tento modelový projekt vhodnější. Tento princip, však je nad rámec výuky 2. ročníku SŠ.



Obrázek 27: Ukázka funkcí k ovládání motoru pomocí H-můstku

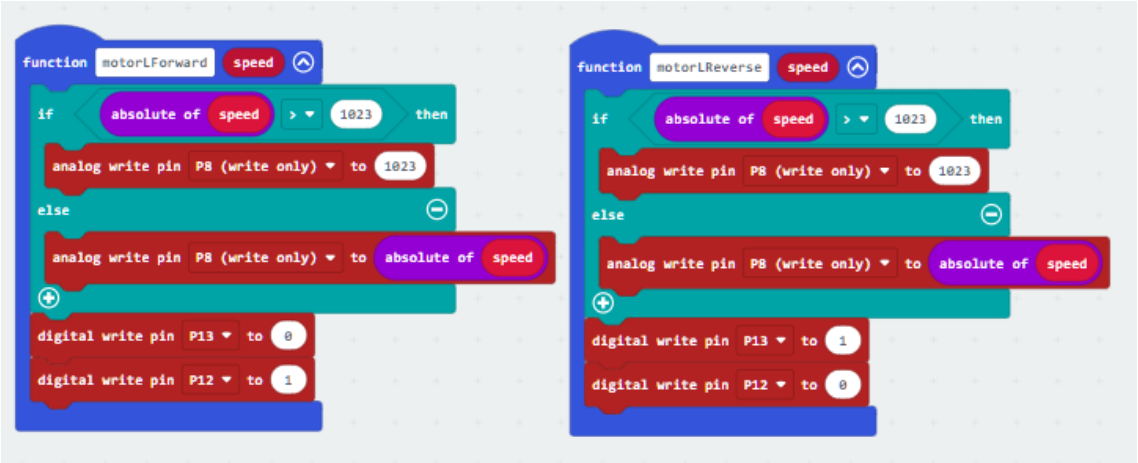
K takto naprogramovaným funkcím byl upraven program *micro:bitu* pro ovládání motorů modelu auta viz. Obrázek 28 , který vychází z Lekce 5 - dálkově řízené vozítko, viz. Příloha 6.



Obrázek 28: Ukázka chyb v programu k ovládání motorů

Po zprovoznění modelu autíčko necouvalo. Bylo nutné si uvědomit, že zatímco u *motor:bitu* se rychlost i směr otáčení řídil kladnou či zápornou hodnotou, zde musí být hodnota vždy kladná. Zapíšeme-li na analogový pin pro řízení rychlosti motoru zápornou hodnotu, *micro:bit* ji nastaví na 0, tím se motor zastaví, čili necouvá. Tato drobná sémantická chyba lze napravit doplněním funkce absolutní hodnoty na proměnnou *speed* (Obrázek 27). Po odstranění této chyby bylo zjištěno, že ovládání modelu funguje v pořádku, až na maximální výchylku joysticku doprava, kdy se má model auta roztočit doprava. Levý motor se postupně roztáčel, ale při dosažení maximální výchylky joysticku

se levý motor zastavil. Díky předchozí chybě, bylo zobrazením hodnoty *speed* rychlosti motoru zjištěno, že proměnná *speed* překročila hodnotu 1023. Pokusem a měřením bylo zjištěno, že *micro:bit* v tomto případě nenastaví analogový výstup na hodnotu 1023, ale na 0, čímž se motor místo jízdy na maximum, zastaví. Tato chyba zřejmě vznikla přepočtem rozsahu joysticku 0 až 1023 na rozsah -100 až 100, který je vysílán z ovladače až po přepočet na rozsah -1023 až 1023 a zaokrouhlením v modelu auta. Problém byl vyřešen otestováním hodnoty *speed* ve funkcích pomocí větvení, viz. Obrázek 29.



Obrázek 29: Ukázka prevence chyb analogového výstupu ovládání motorů.

K počáteční detekci chyb došlo neočekávaným chybným chováním robotické soustavy. Pro žáky by měl být tento typ sémantické chyby poučením, že je důležité kontrolovat hodnoty na konci, tedy v tomto případě na výstupu analogového pinu *micro:bitu*.

Při realizaci projektu je též třeba počítat s tím, že dojde ke zničení elektronických součástí, nejčastěji rozšiřujících desek *micro:bitu* na ovládání motorů. Kontrola zapojení a detekce těchto problémů a následných chyb z nich vzešlých je spíše v kompetenci učitele.

5 Ověření navržených úloh v edukační praxi

Z časových důvodů nebylo možno zařadit ověření navržených úloh do výuky v měsíci září a říjen, jak by odpovídalo původnímu záměru a korespondovalo s výukovým plánem ŠVP [48] předmětu *Úvod do programování*. Reálné nasazení tak bylo posunuto do termínu, kdy vešly v účinnost celostátní opatření s onemocněním COVID-19. Veškerá výuka byla vedena online s velkou tolerancí vůči neaktivitě žáků a s nejasnou vizí návratu k normálnímu režimu. Rozhodl jsem se tedy realizovat ověření úloh s micro:bitem formou online výuky. Z organizačních důvodů, zejména nutného dokončení běžných

projektů dle výukového plánu a klasifikace, jsem ověření úloh zařadil na konec školního roku, tedy dne 24.6.2020 jako závěrečnou hodinu online vyučování. K vedení výuky byly použity online nástroje Microsoft Teams, pomocí kterého byl po souhlasu zúčastněných žáků pořízen video záznam viz.: Příloha 7.

Účast byla s ohledem na podmínky velmi nízká, tato hodina byla vedena jako nepovinná. Část žáků shlédla výuku ze záznamu, což byla též během online výuky běžná praxe. Žáci měli předem k dispozici pracovní listy viz. Příloha 2 až Příloha 4 s doporučením si je vytisknout. Žáci bohužel neměli k dispozici vlastní stavebnice micro:bit, nicméně úlohy jsem upravil tak, aby šlo maximálně využít simulátor micro:bitu v programovacím prostředí *MakeCode* [34]. Dále jsem připravil vhodné videoukázky vlastního micro:bitu, které jsem žákům během online výuky promítal. Žáci též měli k dispozici příslušná videa a materiály v aplikaci Teams, případně mohli využít funkční odkazy na videoukázky. Pro online ukázky micro:bitu jsem použil improvizovaně webkameru jako vizualizér.

Jelikož se k online výuce připojili zejména žáci se zájmem o programování, a také proto, že nebylo nutné prokládat práci s micro:bitem opakováním a korelacemi s programovacím jazykem C#, byly probrány během dvouhodinové výuky všechny tři lekce. Jinak ale byla výuka vedena v intencích přípravy jednotlivých lekcí popsaných v kapitole 4.3. Na konci hodiny jsem žákům ukázal další možnosti k využití micro:bitu, například jak naprogramovat házecí kostku.

5.1 Zpětná vazba z výuky

Po ukončení online hodiny, respektive po shlédnutí videozáznamu, žáci vyplnili krátký dotazník, viz. Tabulka 2. Z dotazníku vyplývá, že všichni žáci výkladu rozuměli. Překvapující je, že někteří žáci, kteří sledovali výuku ze záznamu, nezvládli udělat příklady během hodiny. Nicméně během online vyučování nelze dostatečně posoudit důvody, které vedly k těmto problémům. Ještě překvapivější je zjištění, že více jak polovina žáků nezvládla samostatně najít a opravit chybu v předloženém programu.

Tabulka 2: Dotazník vyplněný žáky po výuce

ID	Ročník – ZŠ/SŠ	Jméno (anonymizované)	Sledoval jsem výuku micro:bit	Souhlasím s tím, že záznam videa bude použit pro další výuku na SPŠE, i v rámci studentských a závěrečných prací na Pedf CUNI. (Bude sdílen jen "kdo má odkaz", nebo přísněji.)	Rozuměl/a jsem výkladu?	Stihl/a jsem udělat všechny příklady v hodině?	Zvládl/a jsem najít a opravit chybu v programu (semaforu) sám/sama.	Uvítal/a bych zařadit práci s micro:bit do výuky?	Líbí se mi micro:bit? [max. 5 hvězd]
1	2.r. SŠ	J. M.	online.	souhlasím	ano	ano	ano	Ne, je to spis výuka pro děti v předškolním věku.	2
2	2.r. SŠ	K. Z.	online.	souhlasím	ano	ano	ano	není to špatné, ale myslím si, že pro nás má větší využití Arduino	3
3	2.r. SŠ	R. M.	online.	souhlasím	ano	ano	ne	ano	4
4	2.r. SŠ	A. N. C.	ze záznamu.		ano	ne	ne	nevím, muselo by se asi zrušit Arduino	4
5	2.r. SŠ	L. P.	ze záznamu.		ano	ne	ne	ano	4
6	2.r. SŠ	M. V.	ze záznamu.		Spíše ano	Jak který	ne	ano	3
7	2.r. SŠ	V. S.	ze záznamu.		ano	ano	ne	ano	4
8	2.r. SŠ	Vik. S.	ze záznamu.		ano	ne	ano	Pokud můžu, tak nebudu utrácet peníze, jinak by to bylo dobrý	5
50	4.r. ZŠ	J. F.	záznam + přímá asistence učitele		ano	ano	ano	ano	5
60	8.-9.r. ZŠ	anonym	Online workshop		ano	ano	ne	ano	5

To vede k závěru, že je třeba s žáky více procvičovat vyhledávání a odladění chyb v algoritmu, což je též cíl této práce. Komentáře ohledně zařazení micro:bit do výuky je třeba bráti z rezervou, neb hodina byla dobrovolná, motivací některých žáků mohlo býti právě porovnání micro:bitu s Arduinem, které probírají v paralelním předmětu. Lekce s micro:bitem z časových důvodů následovaly až po výuce Arduina, oproti původnímu předpokladu, kdy by lekce s micro:bitem na začátku školního roku předcházely výuku

Arduina. Micro:bit by rozhodně nebyl vhodný pro předškolní věk (viz. Tabulka 2 - komentář žáka s ID 1). Nicméně předmětnou video hodinu referenčně absolvoval žák 4. ročníku základní školy s podpůrným přímým vedením učitele a všechny úkoly z lekcí vyřešil[11]. Otázka obliby micro:bitu je subjektivní a netřeba jí komentovat, hodnocení bylo na škále pěti hvězdiček.

Dále byl dne 22.1.2021 realizován online Workshop pro žáky 8. a 9. tříd ZŠ, kde byly realizovány výukové lekce 1-3. Z dotazníkového šetření vyplynulo, že výkladu bylo rozumět, šlo zvládnout časově všechny příklady, bez dopomoci učitele nešlo zvládnout samostatně opravit chybu, je zájem zařadit takovéto lekce do výuky.

Realizovatelnost lekce 4-5 a stavba robotického modelu auta byla vzhledem k distanční výuce ověřena pouze jedním žákem, který jevil o věc zájem.

5.2 Zhodnocení výuky

Takto koncipovaná podpůrná výuka s robotickou programovatelnou pomůckou podporující výuku hlavního programovacího jazyka (C#), jenž probíhá pouze na počítači s frontálním výkladem teorie je vhodná a zvládnutelná nejen v online, ale i v přímé výuce. Vhodné je jí zařadit do hodin určených k opakování látky. Krom zpestření výuky, žáky motivuje, ukazuje jim jinou perspektivu na vytváření algoritmu, pomůže jím lépe pochopit mezipředmětové vazby i vazby na okolní svět. Pomůže žákům více prohloubit a upevnit algoritmické myšlení. Motivuje k používání robotické hračky i pro mimoškolní aktivity.

Dospěl jsem k závěru, že je nutné s žáky procvičovat vyhledávání a odlaďování chyb v algoritmech, byť tato schopnost se lépe procvičuje na větších a komplexnějších projektech než těch navržených v této práci.

V případě přímé, nikoli online, výuky by patrně došlo k opakování stejných hodin v paralelních třídách, a tím pádem k upevnění jistoty učitele v pro výuku komfortnějším prostředí prezenční výuky. Též by došlo k zajištění větší účasti žáků a tím i přesnější, interaktivnější zpětné vazbě od žáků, jak v dotazníku, tak přímo v hodině. Bylo by však nutné zajistit příslušné stavebnice micro:bitu.

[11] U žáků základní školy odpadá korelace se základním programovacím jazykem (C#), avšak je třeba výuku rozdělit do menších celků, neb žáci neudrží tak dlouho pozornost. Též je třeba počítat s tím, že žáci ZŠ budou raději testovat micro:bit na různé vypisování textu a smajlíků, než aby se drželi předmětných úkolů koncipovaných pro SŠ.

Závěr

V této práci jsem v teoretické části zmapoval terminologická východiska, jako algoritmizace, algoritmické myšlení, informatické myšlení. Dále jsem zmapoval metody vhodné pro výuku programování.

Dále jsem vypracoval přehled robotických programovatelných pomůcek vhodných pro využití žáky vyššího sekundárního vzdělávání včetně zhodnocení případného využití v edukativním procesu.

V praktické části jsem připravil, pro již konkrétní zvolenou programovatelnou pomůcku (micro:bit), vhodné výukové lekce s úkoly, které mají podpořit a rozvinou u žáků algoritmické myšlení, práci s chybou a které měly přesně zapadnout a obohatit standardní běžící výuku programování, ve které se dlouhodobě vyskytují u žáků problémy s motivací a pochopením základních principů. Pro výukové lekce jsem připravil pracovní listy, jenž jsou součástí přílohy a krom pokynů a příkladů obsahují odkazy na videoukázky.

V neposlední řadě jsem realizoval, byť online, výuku připravených lekcí a shledal, že je proveditelná. Dospěl jsem k vlastnímu závěru, že právě micro:bit je pomůcka vhodná pro střední školy i bez zaměření na informační technologie a elektrotechniku.

Prací jsem tak splnil hlavní i dílčí cíle, byť cíl ověření v edukační praxi byl omezen a komplikován distanční výukou.

Seznam použitých informačních zdrojů

[1] J. Vaníček, „Výuka algoritmizace patří především do informatiky,“ Počítač ve škole 2016 – celostátní konference učitelů základních a středních škol, Nové Město na Moravě 2016. [Online]. Available: https://www.facebook.com/pocitacveskole/videos /10154798369117782/ sborník: https://docplayer.cz/17569962-Vyuka-algoritmizace-patri-predevsim-do-informatiky.html. [Přístup získán 2020].

[2] M. Svoboda, „ROZVOJ ALGORITMICKÉHO MYŠLENÍ ŽÁKŮ ZŠ VE VÝUCE INFORMATICKY ZAMĚŘENÝCH PŘEDMĚTŮ S VYUŽITÍM SCRATCH,“ UNIVERZITA KARLOVA, PEDAGOGICKÁ FAKULTA, diplomová práce, Praha, 10 09 2018. [Online]. Available: https://dspace.cuni.cz/handle/20.500.11956/104049. [Přístup získán 10 07 2021].

[3] ISTE, „Operational Definition of Computational Thinking for K–12 Education,“ 2011. [Online]. Available: https://id.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2. [Přístup získán 03 07 2020].

[4] J. Fiala, „Rozvoj algoritmického myšlení u žáků druhého stupně základní školy,“ UNIVERZITA KARLOVA, PEDAGOGICKÁ FAKULTA, diplomová práce, Praha, 27 05 2019. [Online]. Available: https://dspace.cuni.cz/handle/20.500.11956/106216. [Přístup získán 10 07 2021].

[5] G. Futschek, „Algorithmic Thinking: The Key for Understanding Computer Science,“ Vienna University of Technology, 2006. [Online]. Available: https://publik.tuwien.ac.at/files/PubDat_140308.pdf. [Přístup získán 05 07 2020].

[6] I. Durdilová, „Principy programování a Programování na SPŠE V Úžlabině 320, Praha 10,“ Střední průmyslová škola elektrotechnická, Praha 10, V

Úžlabině 320, 2012. [Online]. Available: http://uzlabina2.aspone.cz/algoritmuspr.aspx.

[7] VÚP, „Rámcový vzdělávací program pro základní vzdělávání.," Národní ústav pro vzdělávání, 01. 03. 2017. [Online]. Available: http://www.nuv.cz/uploads/RVP_ZV_2017.pdf. [Přístup získán 02. 07. 2020].

[8] ECDL, „Základní moduly ECDL / ICDL," European Certification of Digital Literacy, [Online]. Available: http://www.ecdl.cz/sylaby_basic.php. [Přístup získán 02. 07. 2020].

[9] R. Pecinovský, „Tvorba učebnic a kurzů programování," VŠE v Praze, 2011. [Online]. Available: http://www.vyuka.pecinovsky.cz/prispevky/2011_OB_Tvorba učebnic programování.pdf. [Přístup získán 2020].

[10] „Meet Cubetto, Award winning coding products that help kids age 3+ to learn to code," Primo, [Online]. Available: https://www.primotoys.com/. [Přístup získán 2020].

[11] „Bee-Bot," Terrapin, [Online]. Available: https://www.terrapinlogo.com/products/robots/bee/bee-bot-family.html. [Přístup získán 2020].

[12] Bobřík informatiky, „Informatická soutěž pro žáky základních a středních škol - archiv," Katedra informatiky PF JČU, 2019. [Online]. Available: https://www.ibobr.cz/test/archiv. [Přístup získán 2020].

[13] T. Feltl, „e-Mole č. 11," TFSoft, 11 03 2012. [Online]. Available: https://www.e-mole.cz/system/files/magazine/e-mole_011-2018-mobile.pdf. [Přístup získán 2020].

[14] Přispěvatelé Wikipedie, „Programované učení," Wikipedie: Otevřená encyklopedie., 02 04 2020. [Online]. Available:

https://cs.wikipedia.org/w/index.php?title
=Programované_učení&oldid=18348145. [Přístup získán 2020].

[15] Code, „Naučte se informatiku.," Code, [Online]. Available: https://code.org/. [Přístup získán 2020].

[16] Wendy Hsin-Yuan Huang, Dilip Soman, „A Practitioner's Guide To Gamification Of Education - studie," Rotman School of Management, University of Toronto, 10 12 2013. [Online]. Available: https://inside.rotman.utoronto.ca/behaviouraleconomicsinaction/files/2013/09/GuideGamificationEducationDec2013.pdf. [Přístup získán 2020].

[17] „Socrative," Showbie Inc, [Online]. Available: https://www.socrative.com/. [Přístup získán 2020].

[18] Kahoot!, „Make learning awesome!," Kahoot!, [Online]. Available: https://kahoot.com/. [Přístup získán 2020].

[19] TOGlic, „TOGether Learn In Classroom.," TOGlic, [Online]. Available: https://www.toglic.com/cs/. [Přístup získán 2020].

[20] Duolingo, „Učte se jazyk zdarma. Navždy.," Duolingo, [Online]. Available: https://www.duolingo.com/. [Přístup získán 2020].

[21] Classcraft, „When students are motivated, everyone wins," Classcraft Studios Inc., [Online]. Available: https://www.classcraft.com/. [Přístup získán 2020].

[22] ClassDojo, Inc, „Bring every family into your classroom," ClassDojo, Inc, [Online]. Available: https://www.classdojo.com/. [Přístup získán 2020].

[23] P. Zieleniecová, „Objevování ve škole - heuristická metoda výuky," Matematicko-fyzikální fakultě UK, 11 2012. [Online]. Available: https://kdf.mff.cuni.cz/vyuka/pedagogika/dopl_texty/Heuristicka%20metoda%20vyuky.pdf. [Přístup získán 2020].

[24] J. Dümont, „Programovací nástroje pro algoritmizaci a programováni na ZŠ,“ Univerzita Karlova, Pedagogická fakulta, bakalářská práce, Praha, 09 09 2019. [Online]. Available: https://dspace.cuni.cz/handle/20.500.11956/109410. [Přístup získán 10 07 2021].

[25] „LEGO® MINDSTORMS® Education EV3 Core Set,“ LEGO Education, [Online]. Available: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-core-set/5003400. [Přístup získán 2020].

[26] „LEGO® Education SPIKE ™ Prime,“ LEGO Education, [Online]. Available: https://education.lego.com/en-us/products/lego-education-spike-prime-set/45678#product. [Přístup získán 2020].

[27] „LEGO Education 45678 Spike Prime Základní souprava,“ LEGO Education, [Online]. Available: https://www.alza.cz/hracky/lego-education-45678-spike-prime-zakladni-souprava-d5777319.htm?o=1. [Přístup získán 2020].

[28] „Wex Robotics is designed to easily implement,“ Innovation First International, Inc., [Online]. Available: https://www.vex.com/. [Přístup získán 2020].

[29] „VEX IQ - A STEM Education Revolution,“ Innovation First International, Inc., [Online]. Available: https://www.vexrobotics.com/vexiq. [Přístup získán 2020].

[30] „VEX V5 - The "STEM problem" is real,“ Innovation First International, Inc., [Online]. Available: https://www.vexrobotics.com/vexedr. [Přístup získán 20].

[31] „NABÍDKA ŘEŠENÍ pro vasi školu - ceník,“ AV MEDIA, jaro 2020. [Online]. Available: https://www.avmedia.cz/cs/download/nabidka_reseni.pdf. [Přístup získán 2020].

[32] „Robomaster S1 Learn to Win,“ DJI, [Online]. Available: https://www.dji.com/cz/robomaster-s1. [Přístup získán 2020].

[33] J. Čížek, „Vybrali jsme 21 programovatelných hraček a stavebnic pro děti i jejich rodiče,“ CZECH NEWS CENTER a.s., Živě.cz, 01 01 2020. [Online]. Available: https://www.zive.cz/clanky/nejlepsi-pocitacove-stavebnice/sc-3-a-195794#part=12. [Přístup získán 2020].

[34] „BBC micro:bit,“ Micro:bit Educational Foundation, [Online]. Available: https://microbit.org/. [Přístup získán 2020].

[35] M. Wostl, „micro:bit V1 vs micro:bit V2,“ Zonepi s.r.o., 01 12 2020. [Online]. Available: https://blog.zonepi.cz/microbit-v1-vs-microbit-v2/. [Přístup získán 06 07 2021].

[36] „BBC MICRO:BIT,“ HW kitchen, 2020. [Online]. Available: https://www.hwkitchen.cz/bbc-microbit/.

[37] B. Havířová, „Co je BBC micro:bit,“ Microbiti, [Online]. Available: https://www.microbiti.cz/2019/03/co-je-bbc-microbit.html. [Přístup získán 2020].

[38] „Adafruit MaceCode,“ Adafruit, [Online]. Available: https://makecode.adafruit.com/#. [Přístup získán 2020].

[39] „Code with Mu: a simple Python editor for beginner programmers.,“ Mu, [Online]. Available: https://codewith.mu/. [Přístup získán 2020].

[40] „MicroPython,“ microbit.org, [Online]. Available: https://python.microbit.org/v/2.0. [Přístup získán 2020].

[41] „BBC micro:bit,“ MBED, [Online]. Available: https://os.mbed.com/platforms/Microbit/. [Přístup získán 2020].

[42] „microbit & accessories,“ © Kitronik Ltd, 2020. [Online]. Available: https://kitronik.co.uk/collections/microbit-accessories. [Přístup získán 2020].

[43] „Hover:bit,“ MakeKit, [Online]. Available: https://www.makekit.no/.

[44] „micro:bit Wonder Rugged Car,“ ELECFREAKS, Inc., [Online]. Available: https://www.elecfreaks.com/micro-bit-wonder-rugged-car.html. [Přístup získán 10 07 2020].

[45] „Accessory Guide January 2020,“ © Micro:bit Educational Foundation 2020, 01 2020. [Online]. Available: https://cdn.sanity.io/files/ajwvhvgo/production/c6f0a76cbe149f91fb998fba3f76e4816187e575.pdf?dl=AccessoryGuide.pdf.

[46] „CO JE TO ARDUINO?,“ ARDUINO.CZ, [Online]. Available: https://arduino.cz/co-je-to-arduino/. [Přístup získán 2020].

[47] „Getting started with Raspberry Pi,“ Raspberry Pi Foundation , [Online]. Available: https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started. [Přístup získán 2020].

[48] Zpracoval kolektiv pedagogů SPŠE, Praha 10, V Úžlabině ve školním roce 17/18, „ŠKOLNÍ VZDĚLÁVACÍ PROGRAM,“ Střední průmyslová škola elektrotechnická, Praha 10, V Úžlabině 320, 27 03 2018. [Online]. Available: https://www.uzlabina.cz/uploads/file/SVP_ITE_2018_I1BCD_I2BCD_19_20_minimal.pdf. [Přístup získán 2020].

[49] „Visual Studio 2019,“ Microsoft, 2020. [Online]. Available: https://visualstudio.microsoft.com/cs/vs/. [Přístup získán 07 2020].

[50] „micro:bit Error codes,“ micro:bit, 14 01 2021. [Online]. Available: https://support.microbit.org/support/solutions/articles/19000016969-micro-bit-error-codes. [Přístup získán 08 07 2021].

[51] ELECFREAKS, „motor:bit Smart Car Kit,“ [Online]. Available: https://www.elecfreaks.com/learn-en/microbitKit/motor_bit_smart_car/index.html. [Přístup získán 04 07 2021].

[52] ELECFREAKS Team, „13. Joystick: bit V1,“ ELECFREAKS, 2020. [Online]. Available: https://www.elecfreaks.com/learn-en/microbitExtensionModule/joystick_bit_v1.html?highlight=joystick. [Přístup získán 04 07 2021].

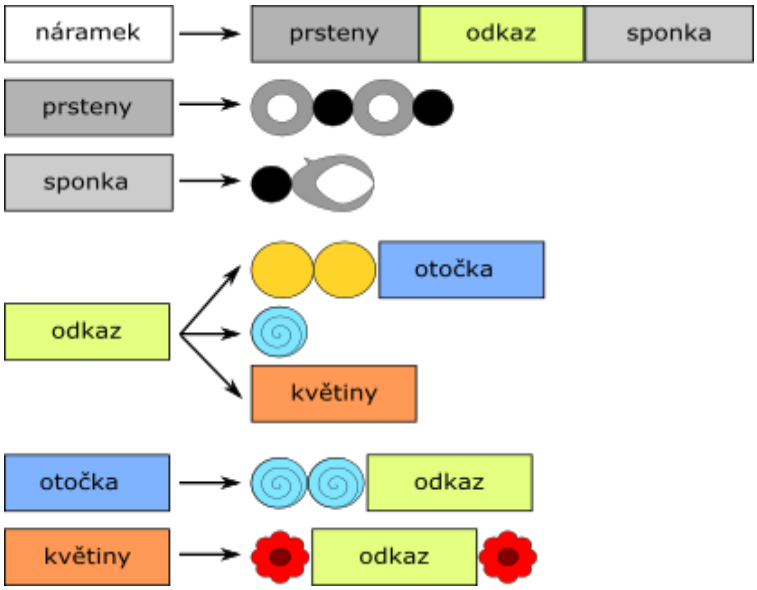
[53] „Motor Driver for micro:bit,“ Waveshare, 17 08 2020. [Online]. Available: https://www.waveshare.com/wiki/Motor_Driver_for_micro:bit. [Přístup získán 06 07 2021].

Přílohy

Příloha 1 – ukázka příkladu ze soutěže Bobřík informatiky

Náramky přátelství

Štěpán vyrábí náramky. Při výrobě postupuje podle pravidel na obrázcích. Každý náramek začíná vytvářet u slova náramek .

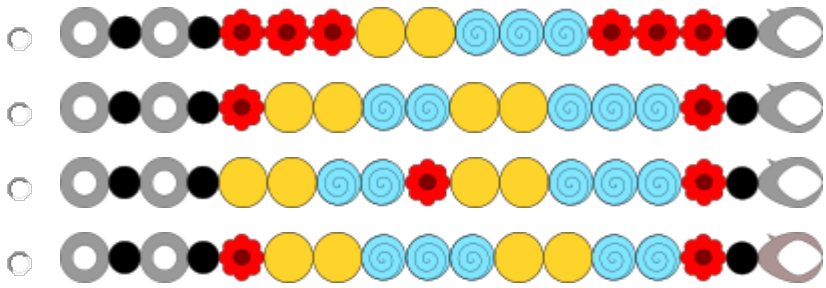


Každý symbol vlevo je nahrazen řadou symbolů, na které odkazuje. Použitím těchto pravidel několikrát za sebou Štěpán vyrobil např. tyto dva náramky:

Štěpán udělal čtyři náramky pro své přátele. Jeden z přátel ale náramek rozbil a jeho majitel pak při jeho opravě udělal chybu.

Dokážeš poznat, který ze čtyř níže uvedených náramků je ten špatně opravený?

Tvoje odpověď:

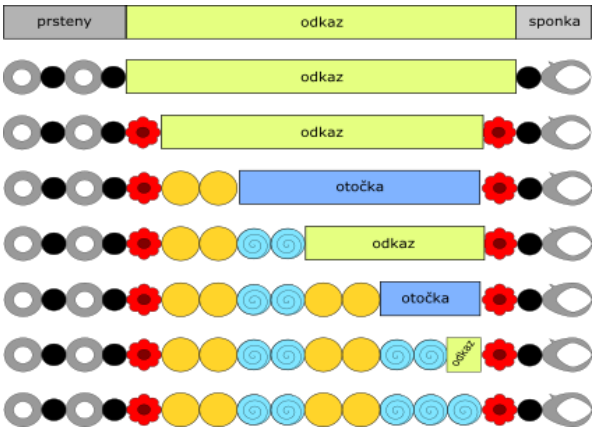


- Nechci odpovídat

Zdůvodnění správné odpovědi

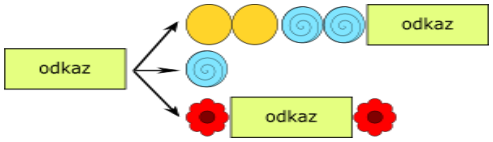
Správná odpověď je

Postupným dosazováním podle pravidel je možné zkontrolovat, zda nějaký náramek lze skutečně vyrobit pomocí Štěpánových pravidel. Na obrázku to ukážeme u jedné odpovědi:



Ukázalo se, že tento náramek lze podle pravidel vytvořit, proto tato odpověď není správná (máme najít nesprávně opravený náramek).

Abychom dokázali, že náramek ze správné odpovědi  pomocí Štěpánových pravidel vyrobit nelze, upravíme pravidla "odkaz", "otočky" a "květiny". Když je pospojujeme do jednoho, vznikne takovéto jednodušší pravidlo:



Jinými slovy, pokud nepoužijeme květinové korálky, pak se každý náramek skládá ze vzoru  , který se opakuje několikrát (možná pouze jednou, nebo dokonce vůbec) a pak musí následovat jeden  . Tedy za třemi modrými korálky nemůže následovat žlutý (může za nimi následovat květina nebo sponka). Toto pravidlo ostatní odpovědi dodržují.

Co má tato úloha společného s informatikou

Soubor pravidel, daný v této úloze, mohl počítač použít ke kontrole, zda Štěpán mohl vyrobit náramek. Podobná pravidla používají počítače ke kontrole, zda programátoři ve svých programech udělali nějaké chyby při psaní, nebo ke kontrole, zda je to, co zadáte do webového formuláře, správné. Příkladem jsou následující pravidla pro kontrolu, zda je nějaký řetězec znaků celé číslo:

- celé číslo → [číslo bez znaménka] nebo [+ číslo bez znaménka] nebo [- číslo bez znaménka]
- číslo bez znaménka → [číslice] nebo [číslo bez znaménka]
- číslice → [0] nebo [1] nebo [2] nebo [3] nebo [4] nebo [5] nebo [6] nebo [7] nebo [8] nebo [9]

Kdybychom toto pravidlo změnili tak, že číslice → [0] nebo [1], měli bychom kontrolu čísla v binárním zápisu (ve dvojkové soustavě).

Tyto typy souborů pravidel v informatice nazýváme gramatika, kontextová gramatika.

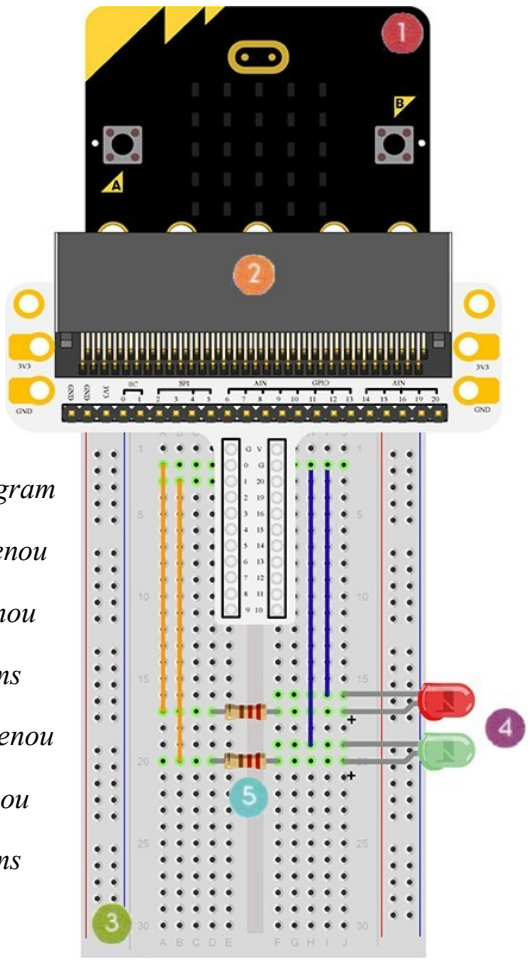
Převzato z: https://www.ibobr.cz/test/archiv

Příloha 2 – Pracovní list lekce 1 - semafor pro chodce

1) Semafor pro chodce:

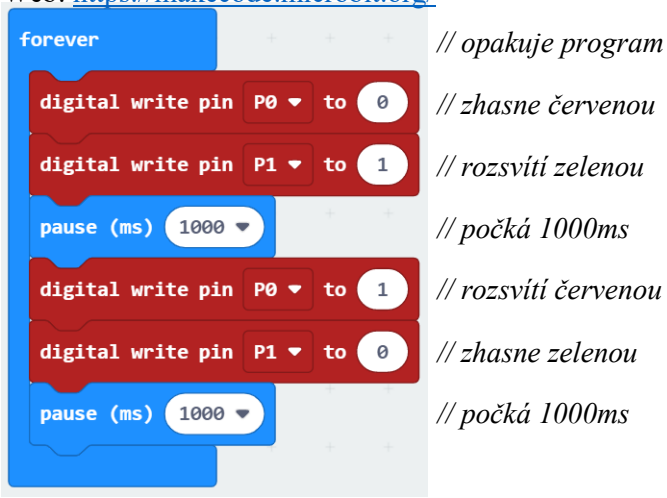
Zapojení

1. 1x deska Micro:bit
2. 1x Micro:bit rozšiřující modul pro kontaktní pole
3. 1x kontaktní pole
4. 1x červená LED, 1x zelená LED
5. 2x rezistor 100Ω



Vytvořme program:

Web: https://makecode.microbit.org/

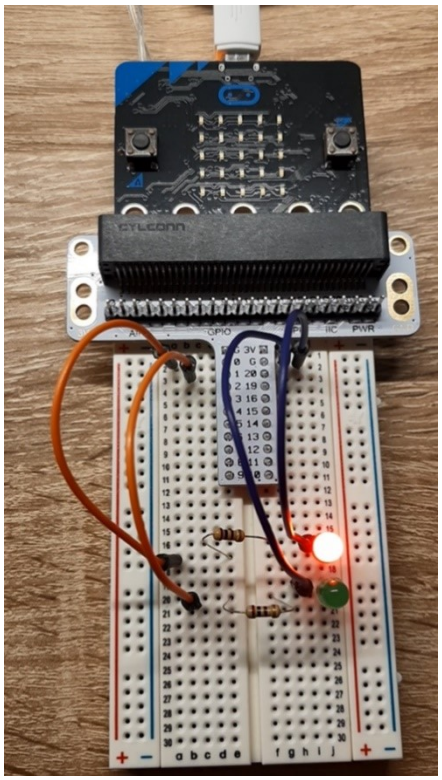


// opakuje program
// zhasne červenou
// rozsvítí zelenou
// počká 1000ms
// rozsvítí červenou
// zhasne zelenou
// počká 1000ms

V cyklu přepínáme pin P0 a P1 a počkáme 1000ms.

Odkaz na stažení programu:
https://1drv.ms/u/s!AkagmpfttUsXvBgNIXpoLl_JpzyV?e=Tu5s6D

Odkaz na videoukázku zapojení:
https://1drv.ms/v/s!AkagmpfttUsXvCP5DAxqHClyCAg1?e=o8bJxk



Rekapitulace:

- Seznámili jsme se zařízením micro:bit.
- Naučili jsme se ovládat digitální piny P0 a P1.
- Vytvořili jsme jednoduchý program a nahráli jej do zařízení.

Zdroje:
https://www.elecfreaks.com/learn-en/microbitKit/Starter_Kit/starter_kit_case_01.html,
https://makecode.microbit.org/

Příloha 3 – Pracovní list lekce 2 – automobilový semafor

2) Úkol – automobilový semafor:

Rozšiřme program a zapojení na tříbarevný semafor pro automobily.

Program by mohl vypadat takto:

Odkaz na stažení programu:
https://1drv.ms/u/s!AkagmpfttUsXvCjX5cwpWgu4_fG6?e=YAdYMP

Posuďte, zda v programu není chyba:

Chová se program podle stavů reálného semaforu? Najděte chybu a program opravte.

Odkaz na video reálného semaforu:
https://1drv.ms/v/s!AkagmpfttUsXvCU8zn04qzk8Yrwn?e=Aa1ynA

Odkaz na videoukázku zapojení:
https://1drv.ms/v/s!AkagmpfttUsXvCyF6sczJF33KXDx?e=bVMvwW

Opravený kód:

Komunikace se studenty, sdílení jejich řešení.

Rekapitulace:

- Pozměnili jsme předchozí program
- Nalezli jsme chybu a opravili jí
- Vyzkoušeli program v micro:bit

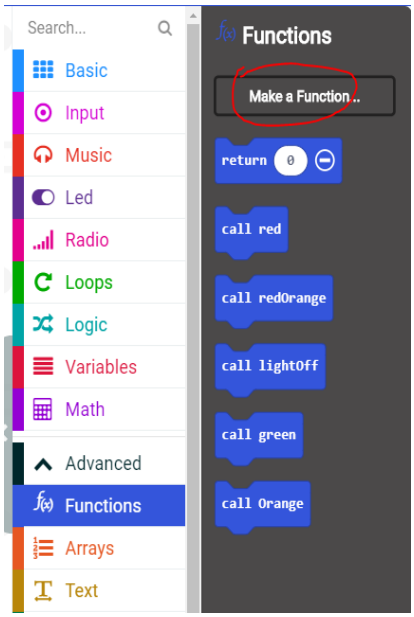
Zdroje:
https://www.elecfreaks.com/learn-en/microbitKit/Starter_Kit/starter_kit_case_01.html,
https://makecode.microbit.org/

Příloha 4 – Pracovní list lekce 3 – větvení a funkce

Funkce

Potřebujeme-li opakovaně vyvolat část kódu, můžeme z této části vytvořit funkci a tu následně volat. Zpřehlední se tak i program.

Funkce vytvoříte pomocí menu: Advanced / Function / Make a Function…

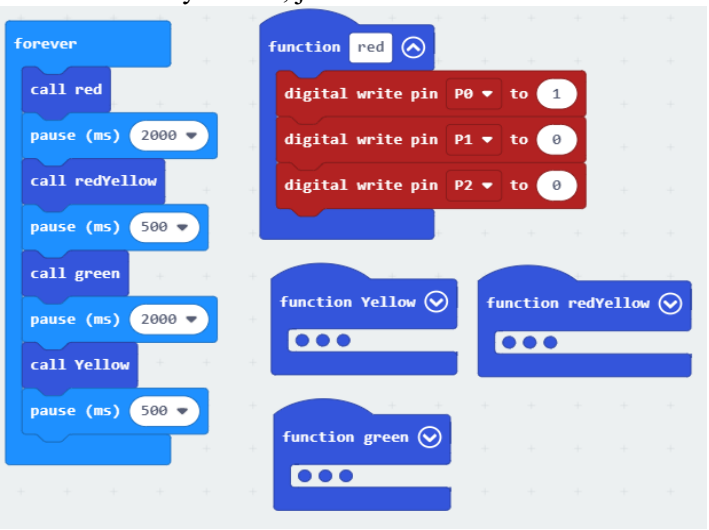


Obr. 1: Menu Funkce

3) Úkol – vytvořte program s funkcemi

Upravte předchozí program semaforu pro automobily, tak aby obsahoval funkce pro rozsvěcení konkrétních světel.

Pozn.: Možná bude stačit přesunout příkazy *digital write* do funkcí. Ale zamyslete se, je to rozumné?

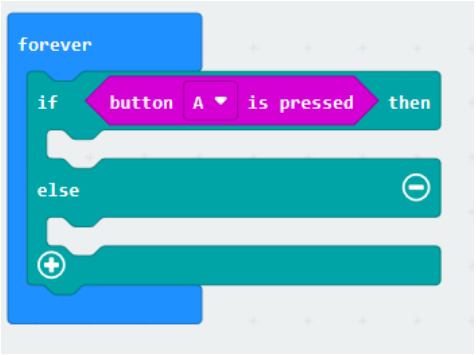


Obr. 2: Příklad programu s funkcemi

4) Úkol – vypnutí semaforu

Nemají-li se auta řídit semaforem, ale značkami, **semafor začne blikat žlutě**. Mohlo by se tak dít automaticky, dle času, nebo podle intenzity dopravy, či slunečního svitu - video: https://1drv.ms/v/s!AkagmpfttUsXvDZHaRG_Z2ObCXak?e=CNMbJz

- Zkuste naprogramovat tento stav semaforu pomocí *funkce Yellow*.
- *V* případě, že budete držet tlačítko *„A“* bude semafor blikat žlutě, pustíte-li tlačítko, začne normálně fungovat.



Obr. 3: větvení a podmínka stisku tlačítka A

Rekapitulace:

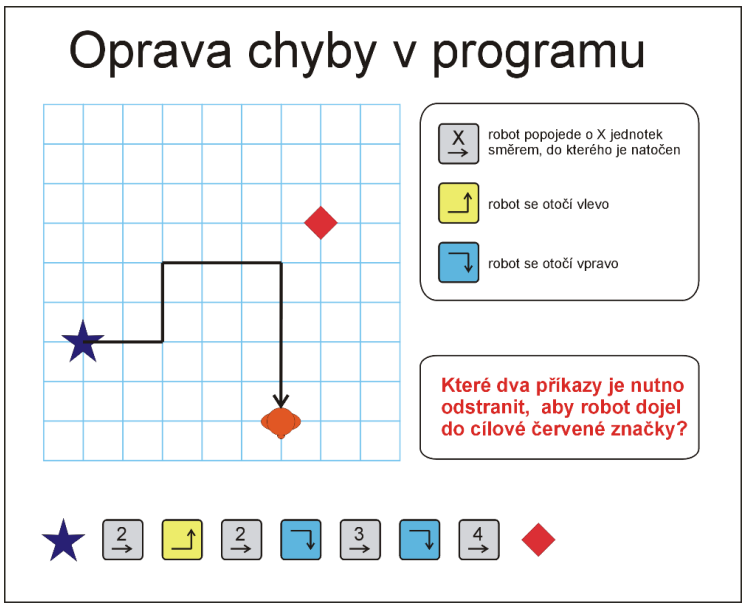
- Naučili jsme se pracovat s funkcemi
- Odladili program a vyzkoušeli
- Naučili se používat tlačítka a větvení

Zdroje:
https://www.microbiti.cz/2020/02/pracovni-listy-funkce-pole.html, https://makecode.microbit.org/

Příloha 5 – Pracovní list lekce 4 – opravte chybu v programu

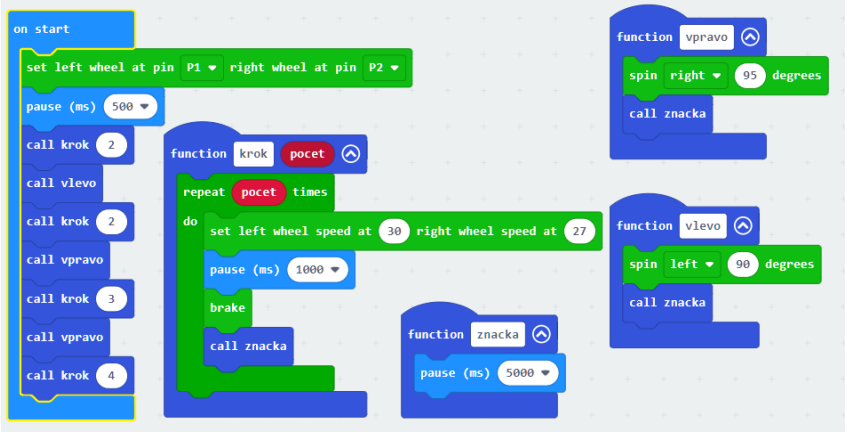
Zadání:

Představte si robota číšníka, který má chybný program, nedojede k červenému stolu, ale jinam.



Obr. 1: Úloha na opravu chybně zadaného algoritmu (převzato z přednášky konference učitelů „Počítač ve škole 2016“)

Zprovozněte robota ring:bit dle programu na obrázku.



Obr. 2: Program robota ring:bit - číšník, který dojede k jinému stolu.

Odkaz pro stažení programu: https://1drv.ms/u/s!AkagmpfttUsXwXLbcpnrqdpJH2CE?e=p0kfdF

Pokyny:

- V programovacím prostředí MakeCode (https://makecode.microbit.org/) přidejte rozšíření *ringbitcar*.
- Nepoužívejte programové bloky  . Robot s kreslící tužkou musí jet pomalu, jinak dochází k překmitům, při rozjezdu, brzdění a otáčení.
- Užívejte bloky  , kde můžete korigovat nestejné servomotory pro jízdu rovně. Maximální rychlost nastavte nejvýše na 30.
- Vyřaďte dva příkazy, tak aby *ring:bit* dojel na správné místo (červenou značku)
- Navrhněte ještě jednodušší program / cestu, kterou je možné také dojet na červenou značku.

Rekapitulace:

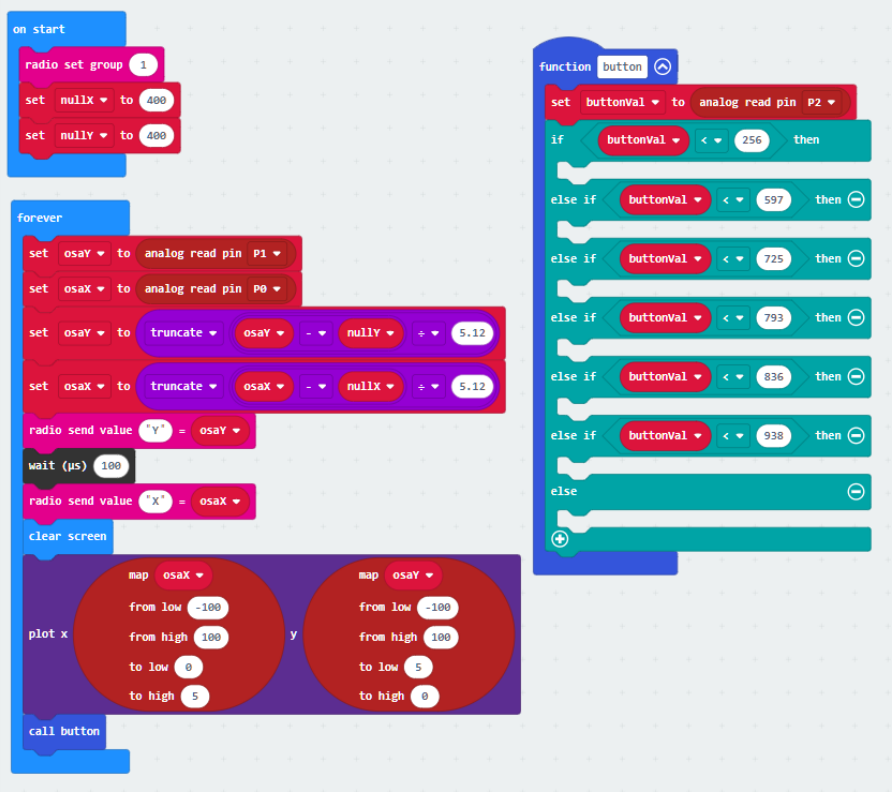
- Seznámili jsme se a zprovoznili jsme programovatelné vozítko Ring:bit.
- Opravili jsme program dle zadání a nalezli ještě optimálnější řešení.

Příloha 6 – Pracovní list lekce 5 - dálkově řízené vozítko

Díky nestejným motorům, pneumatikám, povrchu, síle baterie i joysticku ovladače vozítko motor:bit Smart Car nejezdí rovně, byť by jet mělo. Vozítko případně jede, byť je joystick v klidové poloze.

Program ovladače:

1. **Najděte chyby v programu, tak aby vozítko v klidovém stavu stálo.**
2. **Navrhněte opravu programu tak, aby vozítko jelo rovně. Ideálně tak, aby se dalo nastavení korigovat za jízdy. Navrhněte korekci (tzv. trimy) ovladače v ose X i Y. Pro korekci použijte tlačítka ovladače.**
3. **Pomocí trimu Y rozjeďte vozítko rovně v před.**



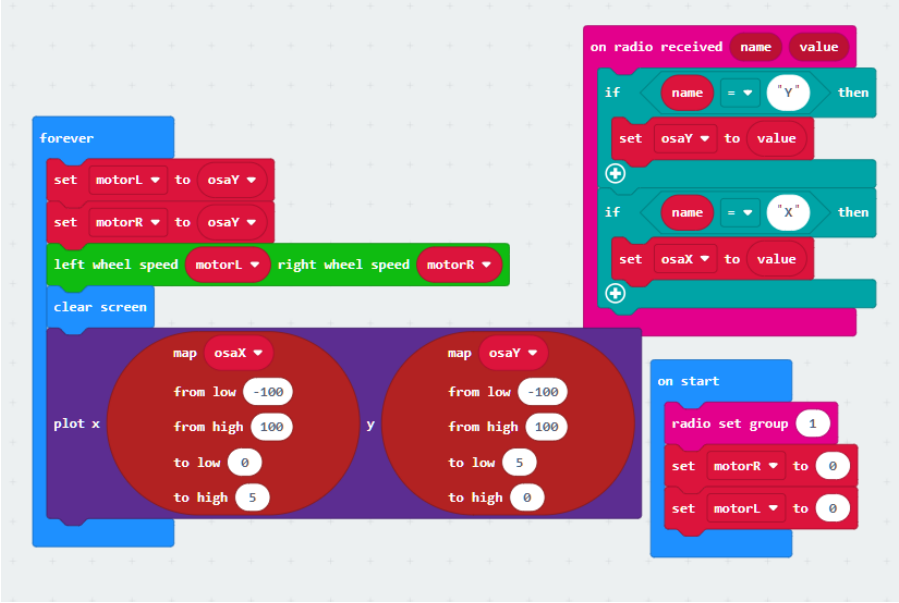
Obr. 1 Základní kód pro ovladač Joystick:bit V1

Web: https://makecode.microbit.org/
Zdroj: https://www.elecfreaks.com/learn-en/microbitExtensionModule/joystick_bit_v1.html?highlight=joystick
Odkaz pro stažení programu: https://1drv.ms/u/s!AkagmpfttUsXwU3ssx2fBIkiNpgu?e=SSj0VK

- Souřadnice XY = [P0,P1] = [0,0] joysticku je vlevo dole. Maximální výchylka XY = [1023,1023] je vpravo nahoře. Pro posun bodu [0,0] na střed je třeba od hodnoty souřadnice odečítat `nullX` a `nullY`. Kolik to přesně má být?
- Souřadnice jsou přepočteny `(osaX-nullY)/5.12` na výchylku -100 až 100. Přepočtené souřadnice se odesílají bezdrátově do druhého micro:bitu.
- Příkaz `plot` vykresluje výchylku joysticku na display. Příkaz map přepočítá výchylku - 100 až 100 na 0 až 5 pro pět LED displeje.
- Funkce `button` rozpozná, které je stisknuto tlačítko A až F. Všechna tlačítka jsou připojena analogově na P2 přes odporový dělič. Každé stisknuté tlačítko přemosťuje jiný rezistor.

Program vozítka motor:bit Smart Car:

4. **Opravte program tak, aby vozítko jezdilo rovně.**
5. **Rozhodněte, zda se korekce má aplikovat na vozítku, či ovladači.**
6. **Navrhněte, jak uplatnit souřadnici X pro jízdu vlevo a vpravo**



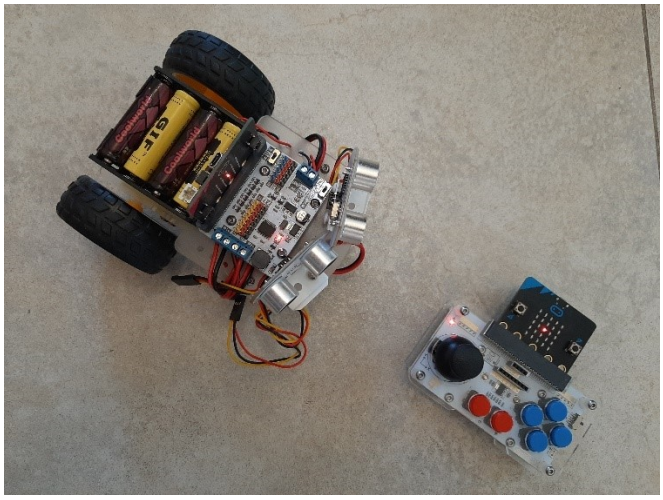
Obr. 2: Základní program pro Motor:bit Smart Car

Odkaz pro stažení programu: https://1drv.ms/u/s!AkagmpfttUsXwVMkOE3Invd-inHc?e=Qq9Plc

- Pomocí události `radio received` přijmeme souřadnice X a Y z joysticku. Zatím používáme pouze souřadnici Y pro jízdu vpřed a zpět.
- Příkaz plot vykresluje výchylku joysticku na display.
- Zelený blok, je příkaz z rozšíření *motorbit*, analogicky lze použít obdobný, např. k *ring:bitu*

Rekapitulace:

- Zprovoznili jsme programovatelné vozítko a dálkově jej ovládáme.
- Odhalili chybu v programu, na jejímž základě vytvořili korekci celé robotické soustavy za běhu programu.
- Naučili se používat bezdrátový přenos mezi micro:bity



Obr. 3: Programovatelné robotické vozítko s micro:bit

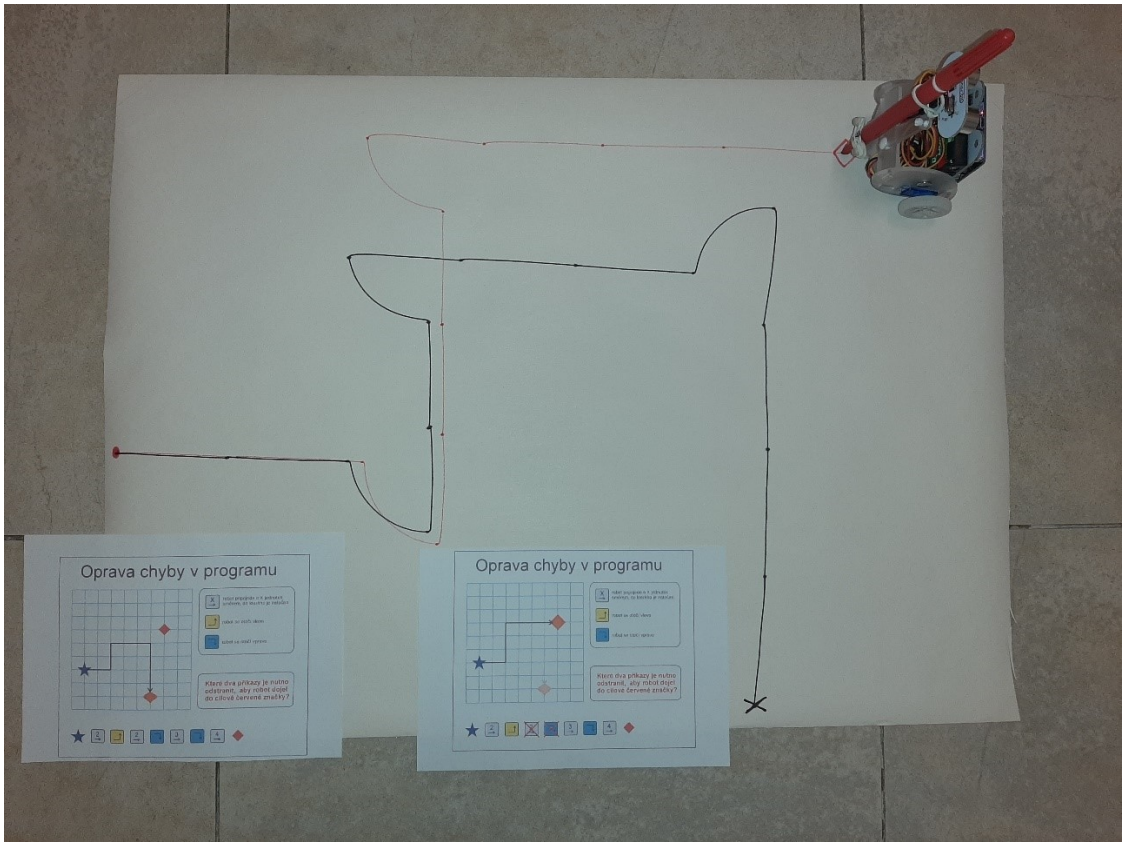
Příloha 7 – Fotodokumentace a videozáznamy

Realizace výuky: záznam výuky Lekce 1-3:

K vedení výuky byly použity online nástroje Microsoft Teams, pomocí kterého byl po souhlasu zúčastněných žáků pořízen video záznam, jenž je dostupný na adrese: https://1drv.ms/v/s!AkagmpfttUsXvEAVvkkjK2ssXL8j?e=pTeuXc.

Realizace lekce: opravte chybu v programu:

Černě vyznačil robot *ring:bit* stopu u chybně zadaného algoritmu. Červeně nakreslil stopu po opravě algoritmu a dojel na zadané místo, viz Obr. 1.

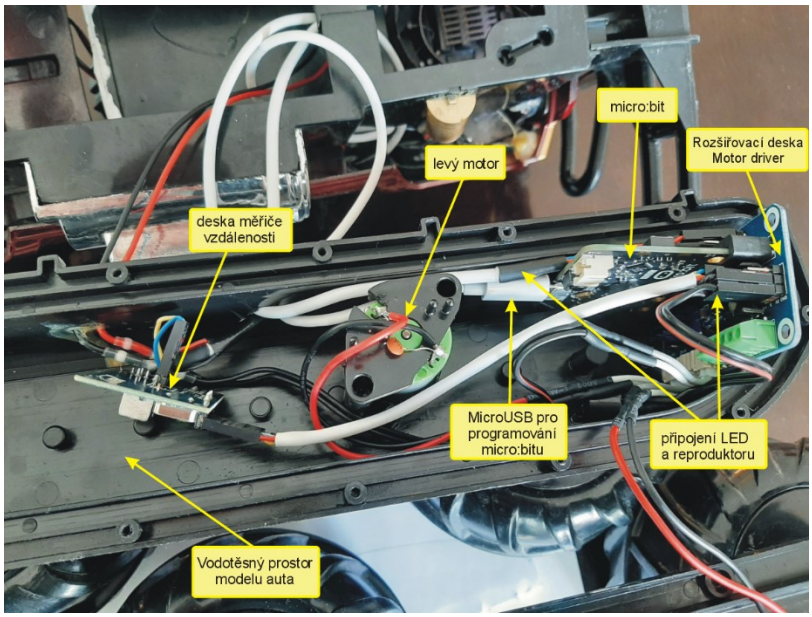


Obr. 1: Ukázka realizace programu robota číšníka a opravy programu.

Realizace projektu robotického modelu auta:

Vybraný model vodotěsného Cruiseru v měřítku 1/10 s instalovaným *micro:bitem* místo originální řídící desky na Obr. 2. s ukázkou vestavby elektroniky na Obr. 3.

Obr. 2: Vodotěsný Cruiser ovládaný micro:bitem.



Obr. 3: Ukázka vestavby micro:bitu do modelu auta.